The Dissertation Committee for Xiaodan Xi
certifies that this is the approved version of the following dissertation:

Modeling Attack Resistant Strong Physical Unclonable Functions: Design and Applications

Committee:

Michael Orshansky, Supervisor

Mohit Tiwari

Nur Touba

Nan Sun

Mudit Bhargava

Modeling Attack Resistant Strong Physical Unclonable Functions: Design and Applications

by

Xiaodan Xi,

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2019

To my beloved family.

Acknowledgments

First and foremost, I would like to thank my supervisor Professor Michael Orshansky for his continuous support and immense guidance over the years. He guided me on how to do independent research rigorously and to express my thoughts effctively to others. I appreciate his creative and critical thinking, academic enthusiasm, and patience that deeply impressed me and made my Ph.D. experience fruitful.

I would like to thank all my committee members: Professor Mohit Tiwari, Professor Nur Touba, Professor Nan Sun, and Dr. Mudit Bhargava for their insightful comments, encouragement, and guidance. My sincere thanks also go to my colleagues and friends during my Ph.D. life: Dr. Ye Wang, Dr. Aydin Aysu, Dr. Meng Li, Dr. Jin Miao, Dr. Jaeyoung Park, Ge Li, Amit Kumar, Ben Ghaemmaghami, Dr. Haoyu Zhuang, Zhuoran Zhao, Chenguang Liu, Mengshi Zhang, Linxiao Shen, Libo Chen, Chunqi Wang, Zhengpeng Hu, and many others for all their help and support in my research and life.

Last but not least, I would like to thank my family for all their love, support, and encouragement. I thank my parents who raised me with all they have and led me to the world of science and technology. I thank my grandparents who protected me and wish me to be happy and healthy forever. I thank my fiancee who gave me love and support during my Ph.D. life.

Modeling Attack Resistant Strong Physical Unclonable Functions: Design and Applications

Publication No. _____

Xiaodan Xi, Ph.D.
The University of Texas at Austin, 2019

Supervisor: Michael Orshansky

Physical unclonable functions (PUFs) have great promise as hardware authentication primitives due to their physical unclonability, high resistance to reverse engineering, and difficulty of mathematical cloning. Strong PUFs are distinguished by an exponentially large number of challenge-response pairs (CRPs), in contrast with weak PUFs that have a smaller CRP set. Because the adversary cannot create an enumeration clone by recording all CRPs even when in physical possession of a PUF, strong PUFs enable secure direct authentication, that does not require cryptography and are thus attractive to low-energy and IoT applications.

The first contribution of this dissertation is the design of a strong silicon PUF resistant to machine learning (ML) attacks. For a strong PUF to be an effective security primitive, the CRPs need to be unpredictable: given a set of known CRPs, it should be difficult to predict the unobserved CRPs. Otherwise, an adversary can succeed in an attack based on building a model of the PUF. Early strong PUFs have shown vulnerability to ML based attacks.

We take advantage of the strongly nonlinear $I-V$ property of MOSFETs operating in subthreshold region to introduce a highly unpredictable PUF. The PUF, termed the subthreshold current array PUF (SCA-PUF), consists of a pair of two-dimensional transistor arrays, a circuit stabilizing the PUF output, and a low-offset comparator. The proposed 65-bit SCA-PUF is fabricated in a 130nm process and allows 2^{65} CRPs. It consumes 68nW and 11pJ/bit while exhibiting high uniqueness, uniformity, and randomness. It achieves bit error rate (BER) of 5.8% for the temperature range of -20 to +80°C and supply voltage variation of ±10%. A calibration-based CRP selection method is developed to improve BER to 0.4% with a 42% loss of CRPs. When subjected to ML attacks, the prediction error stays over 40% on 10^4 training points, which shows negligible loss in PUF unpredictability and about 100X higher resilience than the 65-bit arbiter PUF, 3-XOR PUF, and 3-XOR lightweight PUF.

The second contribution is the application of a strong PUF in a secure key update scheme. Side-channel attacks on cryptographic implementations threaten system security via the loss of the secret key. The adversary can recover the key by analyzing side-channel analog behavior of a cryptographic device, such as power consumption. Fresh re-keying techniques aim to mitigate these attacks by regularly updating the key, so that the side-channel exposure of each key is minimized. Existing key update schemes generate fresh keys by processing a root key using arithmetic operations. Unfortunately, such techniques have been demonstrated to also be vulnerable to side-channel attacks. We propose a novel approach to fresh re-keying that replaces the arithmetic key update function with a strong PUF. We show that the security of our scheme hinges on the resilience of the PUF to a power side-channel attack and propose a realization based on the SCA-PUF. We show that the SCA-PUF is

resistant to simple power analysis and a modeling attack that uses ML on the power side-channel. We target an insecure device and secure server encryption scenario for which we provide an efficient and scalable method of PUF enrollment. Finally, we develop an end-to-end encryption system with PUF-based fresh re-keying, using a reverse fuzzy extractor construction.

The third contribution is the implementation of a strong PUF provably secure against ML attacks. The security is derived from cryptographic hardness of learning decryption functions of semantically secure public-key cryptosystems within the probably approximately correct framework. The proposed PUF, termed the lattice PUF, compactly realizes the decryption function of the learning-with-errors (LWE) public-key cryptosystem as the core block. The lattice PUF is lightweight and fully digital. It is constructed using a weak PUF, as a physically obfuscated key (POK), an LWE decryption function block, a pseudo-random number generator in the form of a linear-feedback shift register (LFSR), a self-incrementing counter, and a control block. The POK provides the secret key of the LWE decryption function. A fuzzy extractor is utilized to ensure stability of the POK. The proposed lattice PUF significantly improves upon a direct implementation of LWE decryption function in terms of challenge transfer cost by exploiting distributional relaxations allowed by recent work in space-efficient LWEs. Specifically, only a small challenge-seed is transmitted while the full-length challenge is re-generated by the LFSR resulting in a $100X$ reduction of communication cost. To prevent an active attack in which arbitrary challenges can be submitted, the value of a self-incrementing counter is embedded into the challenge seed. We construct a lattice PUF that realizes a challenge-response pair space of size 2^{136}, requires 1160 POK bits, and guarantees 128-bit ML resistance. Assuming a bit error rate of 5% for

SRAM-based POK, 6.5K SRAM cells are needed. The PUF shows excellent uniformity, uniqueness, and reliability. We implement the PUF on a Spartan 6 FPGA. It requires only 45 slices for the lattice PUF proper and 233 slices for the fuzzy extractor.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Physical Unclonnable Functions

Physical unclonable functions (PUFs) have great promise as hardware authentication primitives due to their attractive security properties including physical unclonability, high resistance to reverse engineering, and difficulty of mathematical cloning. PUFs are made possible by the imperfections in the fabrication technology that make cloning of a chip with the same device characteristics impossible. As a result, PUFs provide better security compared to conventional key storage based on non-volatile memory that is vulnerable to reverse engineering attacks. PUFs can be used in chip identification, authentication, lightweight encryption, and protection of intellectual property.

Several silicon-based PUFs have been proposed [50, 76, 75, 33, 15, 54, 48, 10, 61, 92, 6]. The amount of entropy in the PUF allows a distinction between strong and weak PUFs. Strong PUFs are defined by an exponentially large number of challenge-response pairs (CRPs), in contrast to weak PUFs that have a small CRP set. Because the adversary cannot create an enumeration clone even when in physical possession of a PUF, strong PUFs enable secure direct authentication that does not require cryptography, and are thus attractive for low-energy and IoT applications.

1.2 Strong PUF based on Subthreshold Current Array

For a strong PUF to be an effective security primitive, the CRPs need to be unpredictable: given a set of known CRPs, it should be difficult to predict the unobserved CRPs. Otherwise, an adversary can succeed in an attack based on building a model of the PUF. Unfortunately, early strong PUFs [50, 76, 74, 92] have shown vulnerability to being attacked through a construction of a mathematical model via machine learning (ML). The most well-known strong PUF is a delay arbiter PUF in which two multi-stage paths with the same nominal delay are set up [50]. Due to variation in device properties, the two paths have slightly different delays and the arbiter identifies a faster path, producing a 1-bit output. However, because a path delay can be expressed as a linear function of individual stage delays, the arbiter PUF can be easily attacked via ML.

Several ways of injecting nonlinear behavior have been proposed [28, 50, 76, 59]. In a feed-forward PUF [50], configuration of the later stages is determined by the results of the earlier stages. Yet, advanced ML attacks utilizing evolutionary optimization can still be effective in building an accurate model [73]. Another way to introduce nonlinearity is to use the XORed result of multiple arbiter outputs as the final PUF output [28]. However, combining outputs from multiple arbiters leads to higher noisiness of the output. And with properly selected model, the XOR PUF can still be successfully attacked (using logistic regression) [73].

The concept of a highly nonlinear ML resistant PUF based on the subthreshold current array (SCA), termed as the SCA-PUF, was proposed by Kalyanaraman *et al.* [39]. The basic building block of SCA-PUF is a pair of nominally identical $n \times k$ two-dimensional transistor arrays with all devices

subject to stochastic variability operating in subthreshold region. The output voltage difference between the two arrays is digitized via a comparator, and thus a PUF response is generated. The principle feature of the circuit is that it has a highly nonlinear boundary between the regions of PUF 1-outputs and 0-outputs in the nk-dimensional space of threshold voltage. This makes existing ML methods fail in predicting the responses of the SCA-PUF which achieves high security.

We present a practical realization of the SCA-PUF [89]. The offset voltage of the comparator is required to be removed through calibration. Since the comparator offset depends on the input common-mode voltage, it is necessary to ensure that the output common-mode voltage of the two SCAs remains constant under different challenge inputs. We designed a common-mode feedback circuit that ensures the comparator to produce a correct result after foreground offset calibration. The comparator is designed based on a Strong-Arm latch and has low noise and offset. Specifically, we report a fabricated strong PUF in 130nm CMOS that has 2^{65} CRPs. When subjected to a suite of ML attacks, the PUF shows resilience that is ~100X higher than that of the 65-bit arbiter PUF, 3-XOR PUF, and 3-XOR lightweight PUF, with negligible loss in PUF unpredictability. When used in conjunction with a reliability-driven CRP selection method, the BER of only 0.4% is achieved for temperature -20 to 80°C and supply voltage ±10%. The SCA-PUF also shows good uniformity and uniqueness.

1.3 Fresh Re-Keying via Strong PUFs

Side-channel attacks are a crucial threat for cryptographic schemes running on embedded devices. The adversary can recover secret information by

analyzing side-channel analog behavior of a device, such as power consumption, during the execution of a cryptographic algorithm [47]. Traditional countermeasures against power-based side-channel attacks include masking and hiding. Masking adds randomness to key-dependent computations to remove correlations between the key and the intermediate values [30], and hiding adds redundant activity into the circuit to achieve constant power consumption [83]. Although these methods improve the security of the system by increasing the number of traces required to extract the key, they do not fully eliminate the side-channel threat [94, 87].

To remove above side-channel vulnerabilities, Medwed *et al.* proposed a new scheme called fresh re-keying that regularly updates the key to limit per-key side-channel exposure [62]. A key update function is used, along with a master key, to generate fresh secret keys for cryprographic primitives. Unfortunately, the adversary can still apply differential power analysis (DPA) on the key update function to extract the master key, and thus obtain all fresh secret keys. Hence, the key update function needs to be carefully designed to be secure against both SPA and DPA attacks. Several re-keying constructions were shown to be vulnerable to such side-channel attacks [67, 84, 20]. These vulnerabilities motivate our search for a fundamentally different approach to designing the key update function.

We propose to replace the algorithmic (mathematical) key update function with a physical object, namely, the PUF, in a way that the construction becomes more secure against the side-channel vulnerabilities of earlier designs [88]. The desired properties are achieved by adopting a strong PUF. For our purposes, such a strong PUF can be abstracted as a hardware hash function with a random key, defined by the unique physical instantiation of process

variation values (e.g., gate delays or threshold voltages). The strong PUF accepts an input challenge and generates responses used to produce a fresh secret key. Specifically, a reverse fuzzy extractor (RFE) is used to produce a high-entropy key along with helper data [21, 85].

Similar to algorithmic re-keying, PUF-based re-keying needs to eliminate the vulnerability to side-channel analysis, such that observing PUF challenges and the power of the PUF does not reveal the secret process variation values. However, Becker *et al.* [8] showed that it is possible to derive those values, for some PUFs, by utilizing power side-channel information. We first hypothesize that a PUF that is constructed to be intrinsically resilient against a CRP-only modeling attack is also resilient against ML attacks utilizing side-channel information. Indeed, if the side-channel does not reveal any information beyond what is already contained in CRPs, then it cannot supply any additional information to the learning algorithm, and thus, a side-channel ML attack is also likely to fail. With this objective in mind, we investigate the use of the SCA-PUF since it is explicitly constructed to be resilient to CRP-based attacks. As our experiments show, it remains secure against the attack used by Becker *et al.* [8].

Because it is undesirable for the server to store PUF CRPs directly, our scheme relies on a model of the PUF. The server performs a one-time extraction of the process variation values during enrollment and uses it later to generate fresh keys. The extraction is made possible via a one-time interface, such as a fuse that is disabled after enrollment, so that the extraction cannot be done after the device is deployed [71, 14]. We provide an enrollment method and evaluate the accuracy of the extracted PUF model.

1.4 A Strong PUF Provably Secure against Machine Learning Attacks

As stated in Section 1.2, in order to be an effective security primitive, a strong PUF needs to be resilient to modeling attacks. The SCA-PUF introduced in Section 1.2 has empirically-demonstrated resistance to some ML algorithms. But empirical demonstrations of ML resistance are not fully satisfactory since they can never rule out the possibility of other more effective ML algorithms. Previous work on ML resistant PUFs [29, 11, 38] also indicates that to design a lightweight provably ML-secure strong PUF is very hard [86]. We propose a strong PUF that is lightweight and provably secure against ML attacks with both classical and quantum computers. The main insight is the mapping of ML attack resistance in a PUF to hardness of learning a decryption function of a cryptosystem within the probably approximately correct (PAC) framework. The proposed PUF is developed based on the earlier proof that PAC-learning a decryption function of a semantically secure public-key cryptosystem entails breaking that cryptosystem [44, 45, 46].

Specifically, we develop a PUF for which the task of modeling is equivalent to PAC-learning the decryption function of a learning-with-errors (LWE) public-key cryptosystem. LWE cryptosystems are based on the hardness of LWE problem that ultimately is reduced to the hardness of several problems on lattices [70]. The LWE decryption function is the core module of the lattice PUF, generating response (plaintext) to each submitted challenge (ciphertext). We develop a measure of ML security in terms of the total number of operations needed to learn the model of the PUF. Such concrete hardness is established by the analysis of state-of-the-art attacks on the LWE cryptosystem [52, 63] and evaluated by the estimator developed by Albrecht *et al.* [3]. Using this

estimator, we say that a PUF has k-bit ML resistance if a successful ML attack requires 2^k operations.

The theoretical security guarantee assumes that the inputs to the LWE decryption function are generated by an encryption function operating on the uniformly random plaintexts: we call such allowed queries "challenges generated by a ciphertext distribution". However, we found that a direct implementation of lattice PUF, in which the server fully generates ciphertext, is inefficient due to the well-known high ratio of ciphtertext to plaintext. We solve this problem by exploiting distributional relaxations allowed by recent work in space-efficient LWEs. First, our approach replaces the component of ciphertext, dominating transmission cost, by a uniformly sampled random vector, such that the resulting distribution is statistically close to the original ciphertext distribution [2]. The advantage of the above replacement is that, as shown by Galbraith *et al.* [25], multiple simple pseudo-random number generators (PRNGs), including those based on a linear-feedback shift register (LFSR), are capable of producing it. Specifically, Galbraith *et al.* shows that input challenges generated by PRNGs provide similar concrete security guarantees against standard attacks on LWE [25]. The proposed strategy allows introducing a low-cost PRNG based on an LFSR and transmitting only a small seed. This results in a dramatic reduction of the effective challenge size. In the improved design with the same parameters chosen above, only 928 bits are needed to produce a 100-bit response. This is a 100X reduction of communication cost in authentication, in contrast to the direct PUF implementation as LWE decryption function.

The focus of the paper is a PUF that is secure against passive attacks in which the observed challenges can be used to derive an internal model of the

PUF. However, the LWE decryption function is vulnerable to an active attack that supplies arbitrary input challenges. (As we show, this risk also carries into an LFSR-based variant). We overcome the risk of such an attack by adopting the technique used by Yu *et al.* [93]: we introduce a self-incrementing counter to embed the counter value into a challenge seed. This makes the attack impossible as the counter restricts the attacker's ability to completely control input challenges to the LWE decryption function.

We implement the PUF on an FPGA which requires a 1160-bit secret key while guaranteeing 128-bit ML resistance. The secret key is generated from POK bits. We use an FE with concatenated codes to reconstruct stable POK bits. The random source of the POK can be any weak PUF. The SRAM PUF power-up states are used in our paper. Assuming an average bit error rate (BER) of 5% for raw SRAM cells, the total number of raw SRAM bits needed is 6.5K, in order to achieve a key reconstruction failure rate of 10^{-6}. The LFSR utilizes a 256-bit seed. The self-incrementing counter produces a 128-bit output. Additional 128 bits are concatenated with the counter output to form the input seed to the LFSR. Thus, the resulting lattice PUF is able to achieve a CRP space of size 2^{136}. The mean BER (intra-class Hamming distance (HD)) is 4.43%. The lattice PUF also shows excellent uniformity and uniqueness. The hardware implementation on a Xilinx Spartan 6 FPGA utilizes only 45 slices for the lattice PUF logic and 233 slices for the concatenation-code-based FE. Compared to several known strong PUFs, the proposed PUF is significantly more resource-efficient.

1.5 Dissertation Outline

The rest of the dissertation is organized as follows: Chapter 2 introduces design details of the SCA-PUF and reports sillicon test results. Chapter 3 discusses the fresh re-keying scheme based on the SCA-PUF. Chapter 4 introduces the theory and design details of the lattice PUF. Chapter 5 concludes this dissertation with a discussion of limitations of the current methods and possible future improvements.

Chapter 2

Strong Subthreshold Current Array PUF Resilient to Machine Learning Attacks

2.1 Introduction

[1] Physically unclonable functions (PUFs) have great promise as hardware authentication primitives due to their attractive security properties including physical unclonability, high resistance to reverse engineering, and difficulty of mathematical cloning. PUFs are made possible by the imperfections in the fabrication technology that make cloning of a chip with the same device characteristics impossible. As a result, PUFs provide better security compared to conventional key storage based on non-volatile memory that is vulnerable to reverse engineering attacks. PUFs can be used in chip identification, authentication, lightweight encryption, and protection of intellectual property.

Several silicon-based PUFs have been proposed [50, 76, 75, 33, 15, 54, 48, 10, 61, 92, 6]. The amount of entropy in the PUF allows a distinction between strong and weak PUFs. Strong PUFs are defined by an exponentially large number of challenge-response pairs (CRPs), in contrast to weak PUFs that have a small CRP set. Because the adversary cannot create an enumeration clone even when in physical possession of a PUF, strong PUFs enable

[1] Contents of this chapter are based on [89] in which the author made substantial contributions to the development and design of the main idea.

secure direct authentication that does not require cryptography, and are thus attractive for low-energy and IoT applications.

However, for a strong PUF to be an effective security primitive, the CRPs need to be unpredictable: given a set of known CRPs, it should be difficult to predict the unobserved CRPs. Otherwise, an adversary can succeed in an attack based on building a model of the PUF. Unfortunately, early strong PUFs [50, 76, 74, 92] have shown vulnerability to being attacked through a construction of a mathematical model via machine learning. The most well-known strong PUF is a delay arbiter PUF in which two multi-stage paths with the same nominal delay are set up [50]. Due to variation in device properties, the two paths have slightly different delays and the arbiter identifies a faster path, producing a 1-bit output. However, because a path delay can be expressed as a linear function of individual stage delays, the arbiter PUF can be easily attacked via machine learning.

Several ways of injecting nonlinear behavior have been proposed [28, 50, 76, 59]. In a feed-forward PUF [50], configuration of the later stages is determined by the results of the earlier stages. Yet, [73] advanced modeling attacks utilizing evolutionary optimization can still be effective in building an accurate model. Another way to introduce nonlinearity is to use the XORed result of multiple arbiter outputs as the final PUF output [28]. However, combining outputs from multiple arbiters leads to higher noisiness of the output. While the XOR PUF is more difficult to learn, with enough computation it can still be attacked (using logistic regression) [73], though the cost of an attack increases exponentially.

The concept of a highly nonlinear machine learning resistant PUF based on the subthreshold current array (SCA) was proposed by Kalyanaraman *et*

al. [39]. In this project, we show a practical design and post-silicon test results of the **SCA-PUF**. In SCA-PUF design, the offset voltage of the comparator is required to be removed through calibration. But the comparator offset depends on the input common-mode voltage, and therefore it is necessary to ensure that the output common-mode voltage of the two SCAs remains constant under different challenge inputs. We designed a common-mode feedback (CMFB) circuit that ensures the comparator to produce a correct result after foreground offset calibration. The comparator is designed based on a Strong-Arm latch and has low noise and offset. Specifically, we report a fabricated 2^{65} CRP strong PUF in 130nm CMOS. When subjected to a suite of machine learning attacks, the PUF shows resilience that is ~100X higher than that of the arbiter PUF, with negligible loss in PUF unpredictability. When used in conjunction with a reliability-driven CRP selection method, the BER of only 0.4% is achieved for temperature -20 to 80°C and supply voltage ±10%. The SCA-PUF also shows good uniformity and uniqueness.

2.2 Subthreshold Current Array PUF: Security Principles and Design

The overall architecture of the proposed strong PUF is shown in Figure 2.1. It consists of two nominally identical two-dimensional transistor arrays selectively subjected to maximum amount of stochastic variability. Each array is composed of k columns and n rows of unit cells, Figure 2.2.

2.2.1 Source of Nonlinearity: Subthreshold Current

Realizing a secure strong silicon PUF requires making the output function nonlinear in random variables. The SCA-PUF utilizes the strongly non-

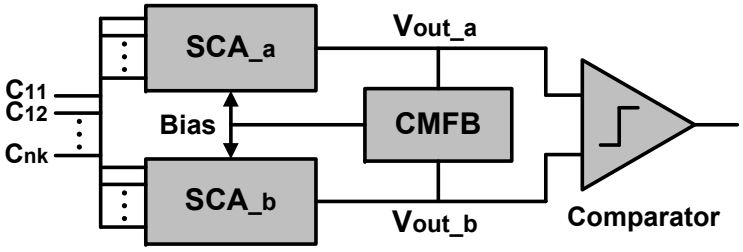


Figure 2.1: The architecture of SCA-PUF consisting of a pair of arrays, a comparator, and a common-mode feedback (CMFB) circuit.

linear I-V behavior of MOSFETs operating in subthreshold region. Consider the equation defining subthreshold current to FET terminal voltages:

$$I_{ds} = \mu C_{ox} \frac{W}{L}(m-1)\left(\frac{kT}{q}\right)^2 e^{\frac{q(V_{gs}-V_{th})}{mkT}}\left(1 - e^{-\frac{qV_{ds}}{kT}}\right) \tag{2.1}$$

where m is the subthreshold slope factor. The critical feature of Equation (2.1) is that the current is exponentially dependent on the threshold voltage V_{th}. Further, the SCA-PUF utilizes the fact that V_{th} exhibits large and spatially-uncorrelated variability due to random dopant fluctuation [81, 65] to ensure uniqueness of individual PUF instances as well as unclonability.

2.2.2 Subthreshold Current Array Structure

A unit cell is composed of two transistors. We use the term stochastic transistor to refer to a device with maximized amount of V_{th} variability which is achieved by using a minimum-sized transistor. In each unit cell, with a row index i and a column index j, a stochastic transistor M_{ij} is in parallel with a non-stochastic switch M_{ijx}, Figure 2.2. The non-stochastic switch is sized to make its variability negligible compared to that of a stochastic device. The bias of each array is chosen such that regardless of the input controls, the currents in each branch keep the diode-connected stochastic transistors in

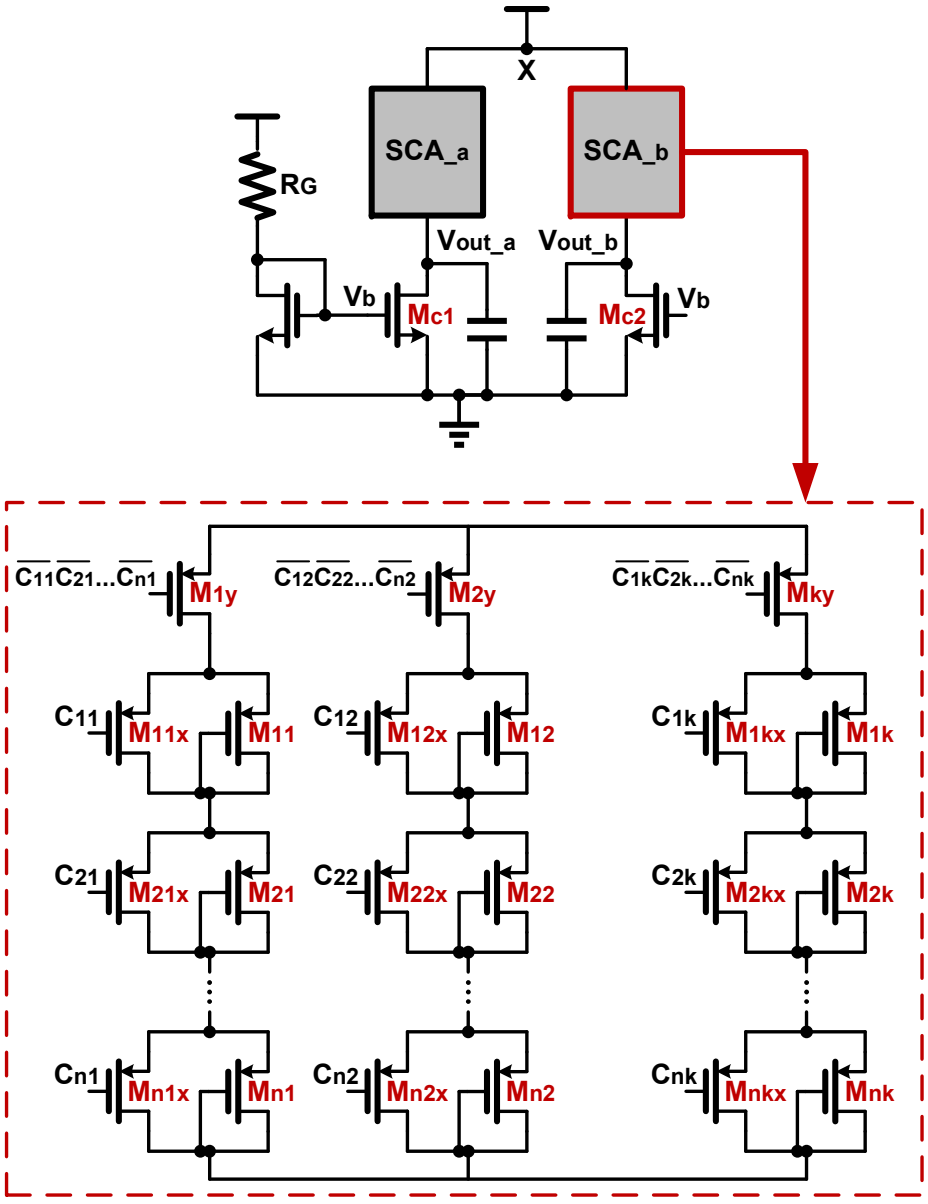


Figure 2.2: The subthreshold current array consists of controllable unit cells.

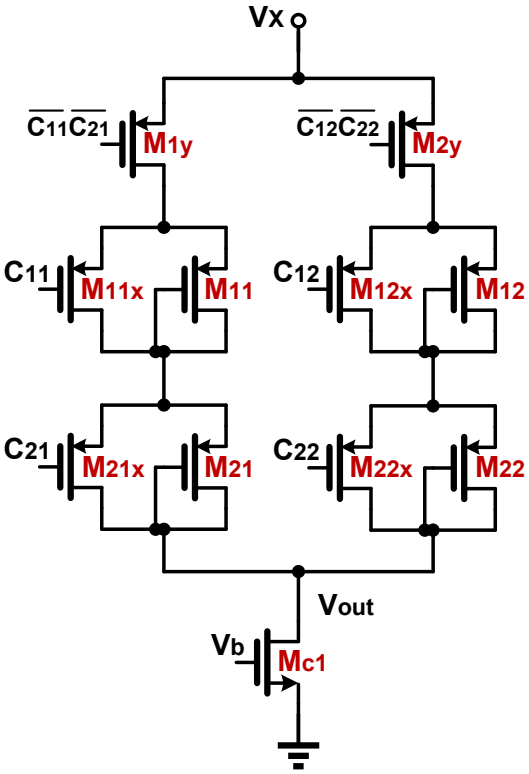


Figure 2.3: An example 2×2 SCA.

subthreshold region. The subthreshold region is ensured via current sources M_{c1} and M_{c2}. When $C_{ij} = 0$, the stochastic transistor is shorted by the switch which removes its impact on the output voltage. When $C_{ij} = 1$, the stochastic transistor operates in subthreshold region and contributes to determining the output voltage. A header transistor M_{jy} is at the top of each column, avoiding a direct up-down path when all the switches are shorted in one column.

The currents through M_{c1} and M_{c2} are equal and constant. Because both arrays are driven by the same input bits C_{ij}, in the absence of device mismatch, the two arrays produce identical output voltages. However, the randomness of V_{th} leads to the difference between the output voltages which

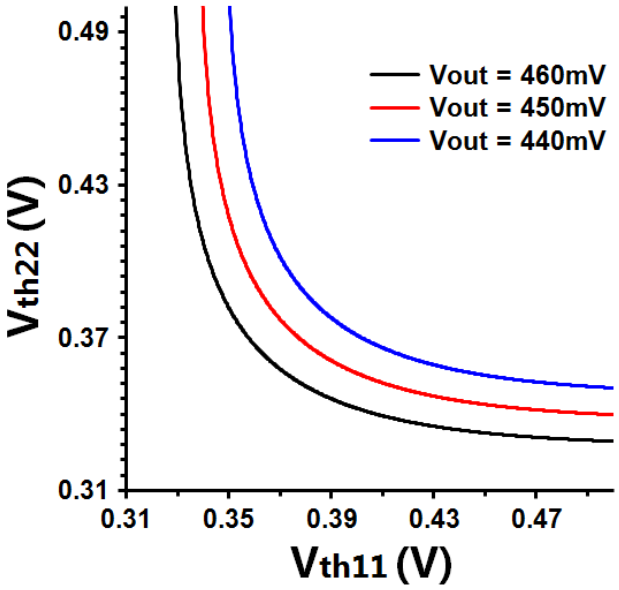


Figure 2.4: A nonlinear curve of constant V_{out} for the 2×2 SCA-PUF produces the hard-to-learn behavior.

is converted to a binary response by the comparator. Because all values of C_{ij} are allowed, the size of the CRP space is 2^{nk}, making it a strong PUF.

The defining characteristic of the SCA-PUF is the highly nonlinear boundary between the regions of PUF 1-outputs and 0-outputs in the nk-dimensional space of V_{th}. This makes existing machine learning methods fail in predicting the response of the PUF. We use a 2×2 SCA of Figure 2.3 for a concrete example. Utilizing Equation (2.1) to analyze Figure 2.3 several curves of constant output voltage in terms of threshold voltages V_{thij} are produced, Figure 2.4. The figure shows a significant nonlinearity of output voltage. Because the degree of nonlinearity may be hard to appreciate visually, we also compute the derivative of the curve in Figure 2.4:

$$dV_{th22}/dV_{th11} \approx -e^{\frac{V_{th12}+V_{th22}-V_{th11}-V_{th21}}{2kT/q}} \tag{2.2}$$

It can be clearly seen that even the slope of the hypersurface varies exponentially with V_{th}. It is this behavior that makes the SCA-PUF resilient to modeling attacks, as we empirically demonstrate later.

2.2.3 Common-Mode Feedback Circuit

As shown in Figure 2.1, the output voltage difference between the two arrays is digitized via a comparator. The offset voltage requirements on the comparator are quite stringent, as detailed in the next section, requiring the offset to be removed through calibration. Since the comparator offset depends on the input common-mode voltage, it is necessary to ensure that the output common-mode voltage of the two SCAs remains constant under different challenge inputs. Otherwise, even if the comparator is calibrated and offset-free at a given common-mode voltage, the offset would re-appear as the common-mode voltage varies.

In the original PUF circuit of Figure 2.2, the node X is directly connected to V_{DD}. The common-mode voltage $V_{out\_CM} \equiv (V_{out\_a} + V_{out\_b})/2$ is highly sensitive to the SCA input C_{ij}, as it alters the pull-up strength of the SCAs, causing V_{out_CM} to vary from 300 mV to 900 mV. Such a large variation makes the comparator offset calibration ineffective. To maintain a stable V_{out_CM}, we insert M_{f5} to separate the node X from V_{DD} and control its gate voltage V_f by a common-mode feedback (CMFB) circuit, Figure 2.5. If V_{out_CM} deviates from the desired common-mode voltage V_{ref}, the error amplifier adjusts the pull-up strength of the SCAs by changing V_f to keep V_{out_CM} close to V_{ref}.

The simplest way to sense the common-mode voltage is to use resistor averaging [68]. This method is not suitable here because the SCAs operate

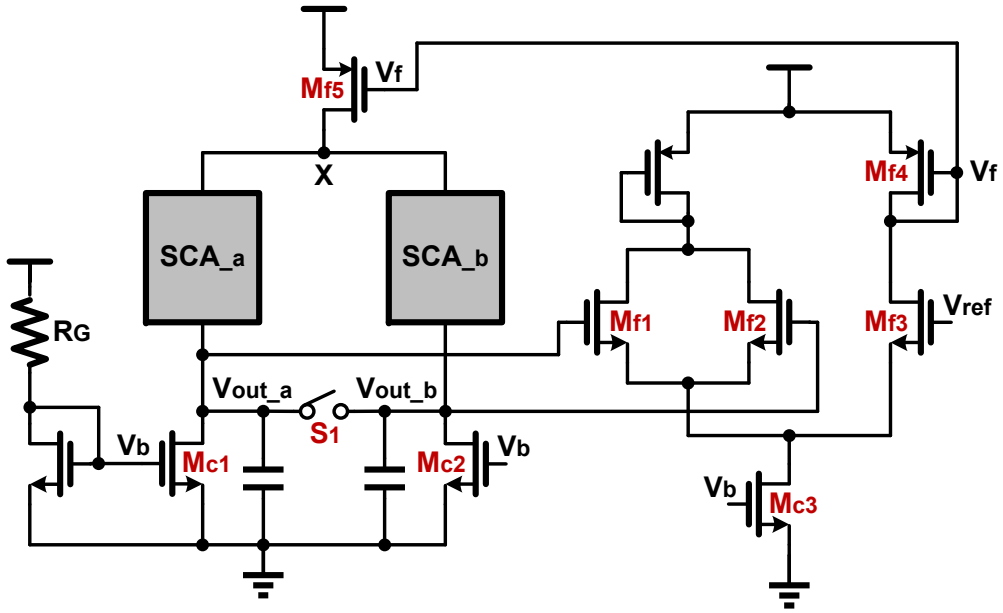


Figure 2.5: Complete PUF circuit with common-mode feedback.

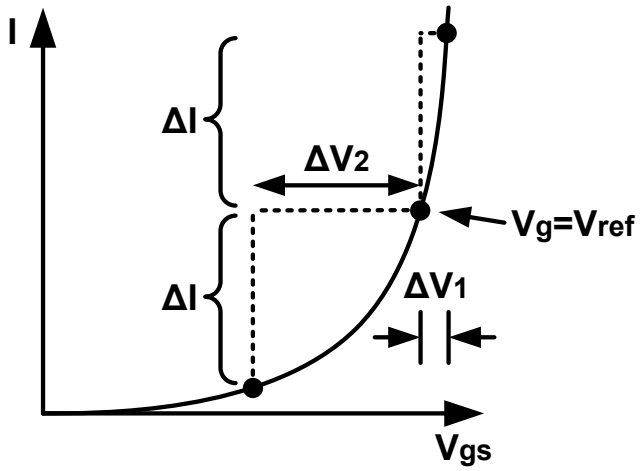


Figure 2.6: $I - V$ curve of MOSFET.

in the subthreshold region with low bias current and high output resistance. Therefore, we would need very large resistors (in GΩ) to avoid attenuating the SCA differential-mode output signal $\Delta V_{out} \equiv V_{out\_a} - V_{out\_b}$. Such a solution would be costly in terms of chip area. To obviate this problem, we choose to use transistors M_{f1} and M_{f2} to perform the current averaging as shown in Figure 2.5. The merit of this approach is that because the input impedance of M_{f1} and M_{f2} is very high, the differential output ΔV_{out} is not attenuated.

We should also note that M_{f1} and M_{f2} act as an accurate averager only when ΔV_{out} is small because of the intrinsic nonlinearity of I-V_{gs} of a transistor. When $\Delta V_{out} = 0$, the CMFB circuit makes V_{out_CM} equal to V_{ref}, where the comparator is well calibrated. When ΔV_{out} is a small value, V_{out_CM} equal to V_{ref} is still guaranteed. However, when ΔV_{out} is sufficiently large (small signal analysis is no longer valid), V_{out_CM} will deviate from V_{ref}. This deviation can be calculated using the MOSFET I-V curve in Figure 2.6:

$$V_{g\_f1} = V_{out\_a} = V_{ref} + \Delta V_1 \tag{2.3}$$

$$V_{g\_f2} = V_{out\_b} = V_{ref} - \Delta V_2 \tag{2.4}$$

$$\Delta V_1 \ll \Delta V_2 \tag{2.5}$$

where V_g is gate voltage. As a result,

$$V_{out\_CM} = \frac{V_{out\_a} + V_{out\_b}}{2} < V_{ref} \tag{2.6}$$

This equation shows how V_{out_CM} deviates from V_{ref}. According to the measurement result, when $\Delta V_{out} = 50\text{mV}$, V_{out_CM} deviates by about 10mV, making the comparator offset (several mV) re-appear. But this comparator offset is much smaller than 50mV ΔV_{out}, so the comparison result is still correct. In summary, the proposed CMFB circuit ensures that the comparator produces a correct result after foreground offset calibration.

2.2.4 Comparator

The PUF response can be influenced by the offset and noise of the comparator. A large offset would significantly deteriorate randomness and inter-die Hamming distance (HD) of the PUF while large noise would degrade the intra-die HD. The allowable offset and noise of the comparator depend on the statistics of the comparator's input differential voltage. The measured differential SCA output voltage $|\Delta V_{out}|$ has the distribution of $\mu = 17.9$ mV and $\sigma = 13.6$ mV (Figure 2.10). Therefore, the input offset V_{os} of the comparator needs to be below 100 μV to ensure that 99.6% of the CRPs are correctly resolved. Similarly, the input referred rms noise should be below 100 μV.

The proposed comparator is based on a strong-arm latch with some modifications, as shown in Figure 2.7. The tail transistor which connects to the clock in the Strong-Arm latch is replaced by M_5 and M_6. Since the source nodes of the input transistors M_1 and M_2 are always connected to the ground, the kick-back noise of the proposed comparator is one half of the noise of the Strong-Arm latch. Although the kick-back noise is reduced, the drain nodes of M_1 and M_2 are at ground during the comparator reset period. Therefore, when the comparator is fired on the rising edge of the clock, M_1 and M_2 may operate in the linear region, causing a large comparator offset and noise [69]. To address this issue, two large capacitors C_1 and C_2 are added. During reset, the capacitors C_1 and C_2 are discharged. When the comparator clock CK_{comp} rises, both V_P and V_Q are pulled to a high voltage V_{max} via capacitive coupling of C_1 and C_2, as shown in Figure 2.8. This ensures that M_1 and M_2 work in the saturation region, which reduces the comparator offset and noise. Moreover, C_1 and C_2 elongate the dynamic integration time of the input transistors, further reducing comparator noise.

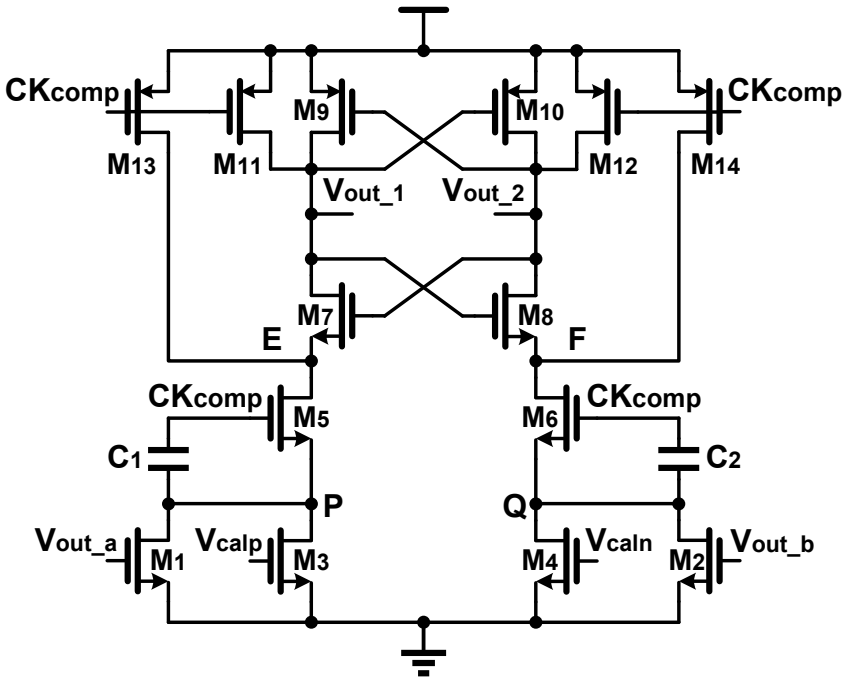


Figure 2.7: Proposed comparator with low offset and noise.

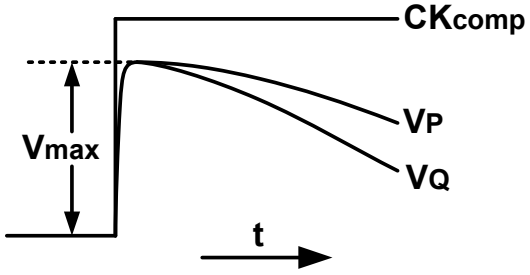


Figure 2.8: Voltages at nodes P and Q in comparator.

In summary, the proposed comparator has three operation phases:

- Reset phase: CK_{comp} is 0; nodes P and Q are reset to GND; and voltages across C_1 and C_2 are 0.
- Amplification phase: V_P and V_Q follow CK_{comp} to go high and reach V_{max}. After that, V_P and V_Q go down slowly, because M_1 and M_2 draw current from P and Q. Simultaneously, V_E and V_F also go down slowly, because M_5 and M_6 are turned on.
- Regeneration phase: The comparator works in the same way as the Strong-Arm latch.

To calibrate the comparator offset, transistors M_3 and M_4 are adopted. The offset calibration process is as follows. We first turn on switch S_1 in Figure 2.5 to ensure $\Delta V_{out} = 0$. Then, we adjust V_{calp} and V_{caln} to ensure that the comparator output topples between 1 and 0 (i.e., stays close to the metastability region).

2.3 Test Chip Measurement Results

The test chips of the SCA-PUF have been fabricated in the IBM 130nm CMOS process. Figure 2.9 shows the die photograph. Each chip contains three 65-bit-input PUF instances.

2.3.1 Analog Outputs Measurement

We first measure the distribution of $|\Delta V_{out}|$, which is directly related to the PUF robustness. When $|\Delta V_{out}|$ is large, the PUF response is less sensitive to the comparator offset and noise, thus improving PUF robustness. $|\Delta V_{out}|$

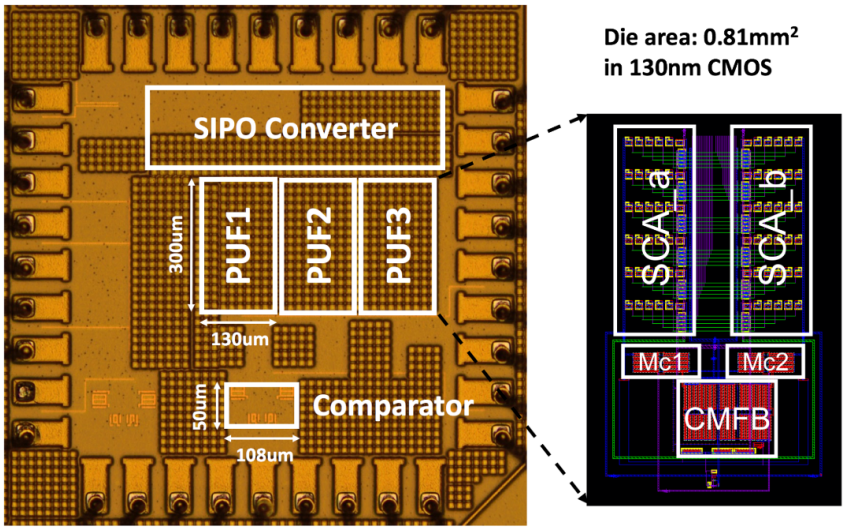


Figure 2.9: Chip micrograph and layout.

is measured across all the 50 PUFs on 17 chips. Figure 2.10 shows that the measured $|\Delta V_{out}|$ distribution has $\mu = 17.9$mV, $\sigma = 13.6$mV. Because our comparator has low offset of less than 100μV after calibration, only 0.35% of the CRPs are impacted.

We also investigate how V_{out_CM} varies with $|\Delta V_{out}|$. This is important because V_{out_CM} can affect the comparator offset. V_{out_CM} vs. $|\Delta V_{out}|$ is measured for 500 challenges for one PUF instance; the results are in the scatter plot of Figure 2.11. When $|\Delta V_{out}|$ is zero, V_{out_CM} is about 463mV: this is the point at which the comparator is calibrated. When $|\Delta V_{out}|$ is 50mV, V_{out_CM} deviates from 463mV by about 10mV: as explained above, this small deviation will not produce an incorrect result.

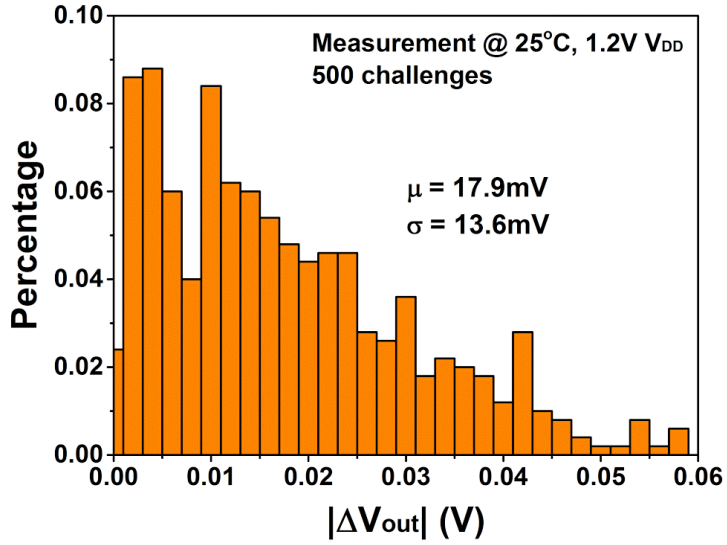


Figure 2.10: Output voltage difference distribution.

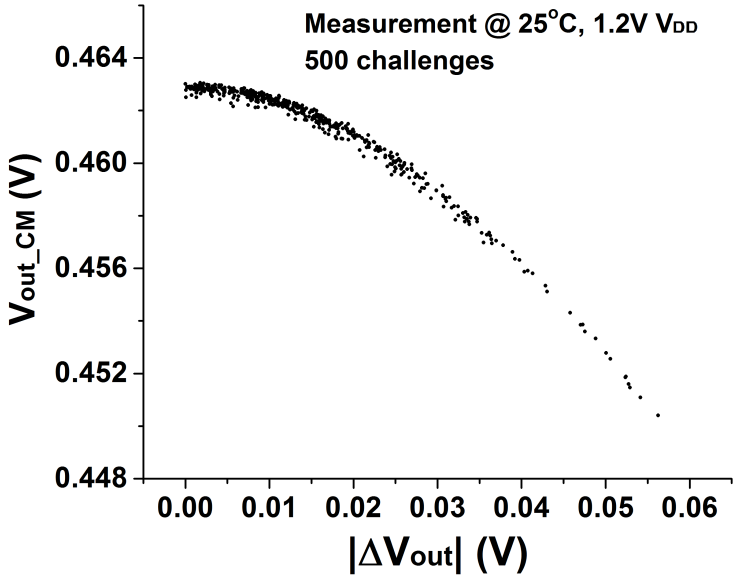


Figure 2.11: Dependence of V_{out_CM} on $|\Delta V_{out}|$ in a single PUF instance.

2.3.2 PUF Robustness Measurements

We first investigate the impact of noise by repeatedly measuring PUF outputs under nominal conditions (25°C and $V_{DD} = 1.2V$). As shown in Figure 2.10, $|\Delta V_{out}|$ can be as small as several μV, which makes only a very small fraction of PUF responses sensitive to noise (e.g. for $|\Delta V_{out}| < 100\mu V$, the fraction of CRPs is 0.35%). We randomly choose 200 CRPs from one PUF instance to evaluate response 100 times. The measurement result shows that only 1 of 200 response bits is unstable.

We then investigate the PUF response robustness against V_{DD} variation. The comparator is calibrated only once under the nominal conditions. Bit error rate (BER) is used to quantify the PUF robustness. In what follows, BER is defined identically with intra-die HD, measuring a fraction of response bits of a PUF that change across different environmental conditions. The supply voltage is varied 1.08 to 1.32V V_{DD} with $V_{DD} = 1.2V$ as a reference. As shown in Figure 2.12, the BER measured for 5 PUFs and 500 challenges is less than 4% in the worst case.

Next we investigate the robustness against temperature variation. The comparator is also calibrated only once under the nominal conditions. As shown in Figure 2.12, the measured BER is 12% in the worst case, across -20~80°C for 5 PUFs and 500 challenges. This indicates that our PUF is more sensitive to temperature than V_{DD}, as further illustrated in Figure 2.12. The overall BER across both -20~80°C and 1.08~1.32V V_{DD} is also measured. The result is shown in Figure 2.14: the average BER (intra-die HD) is μ =0.058 and the standard deviation is σ =0.038.

One effective way to improve PUF robustness is to identify unstable

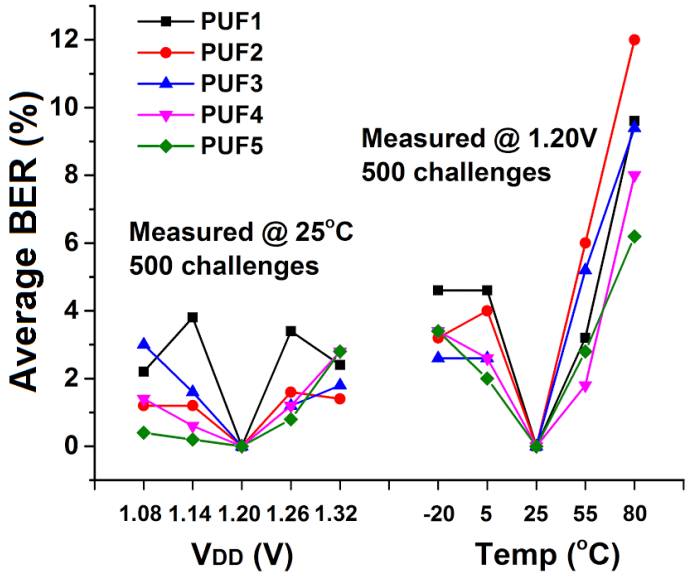


Figure 2.12: PUF robustness against V_{DD} and temperature.

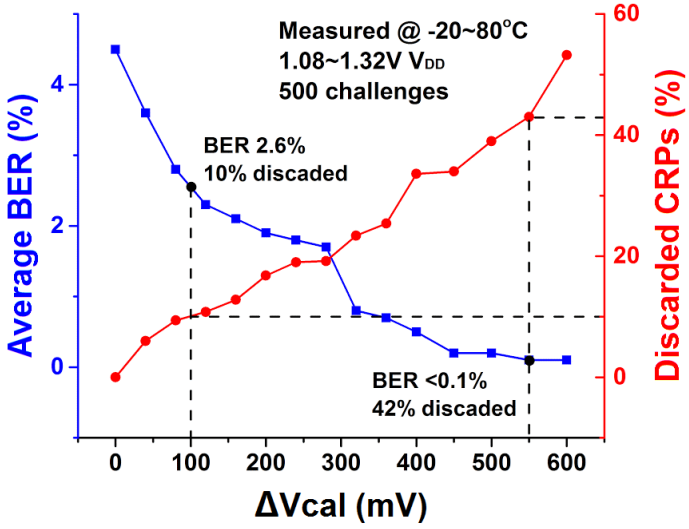


Figure 2.13: BER and the fraction of discarded CRPs at different calibration voltages ΔV_{cal}.

bits and discard (mask) them. The masking information can be stored in nonvolatile memory (NVM). Because the unstable bits are characterized by small $|\Delta V_{out}|$, the value of $|\Delta V_{out}|$ can be used for masking, with bits below a threshold being masked. However, it is difficult to directly measure $|\Delta V_{out}|$ in real applications since adding pins for SCA analog outputs is undesirable.

We propose a solution that uses comparator offset as the indirect way of identifying unstable bits. First, for any given set of challenges, the reference responses are identified as responses produced by the optimally-calibrated comparator. Such a comparator is characterized by a pair of calibration voltages V_{caln} and V_{calp}, see Figure 7. Next, while V_{calp} is unchanged, the responses are recorded again after modifying only V_{caln} by ΔV_{cal}. All bits that have changed compared to the reference set are marked as unstable (discarded). The procedure is repeated, this time keeping V_{caln} at its optimal setting, and modifying only V_{calp} by ΔV_{cal}. All bits that have flipped in this case are also marked as unstable (discarded). Figure 2.13 shows the average BER and the fraction of discarded CRPs as a function of ΔV_{cal}. As ΔV_{cal} increases, more CRPs are marked as unstable and discarded which reduces the average BER. The experimental results indicate that our method can greatly reduce the average BER to 2.6% with a 10% loss of CRPs or 0.1% (0.4% in the worst case) with a 42% loss of CRPs. Since the 65-bit PUF has 2^{65} available CRPs, even if 42% of all CRPs are discarded, we still have $\sim 2^{64}$ CRPs to use.

2.3.3 Uniqueness, Uniformity and Randomness Measurement

Several metrics are widely used to ascertain the quality of a PUF [57]. Ideally, each PUF produces a unique set of responses to the same challenge set. Uniqueness, that is typically quantified as inter-die HD, is the measure

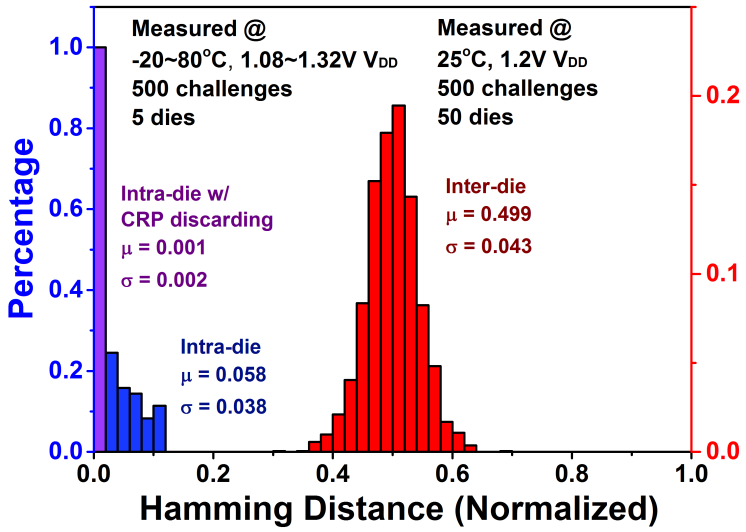


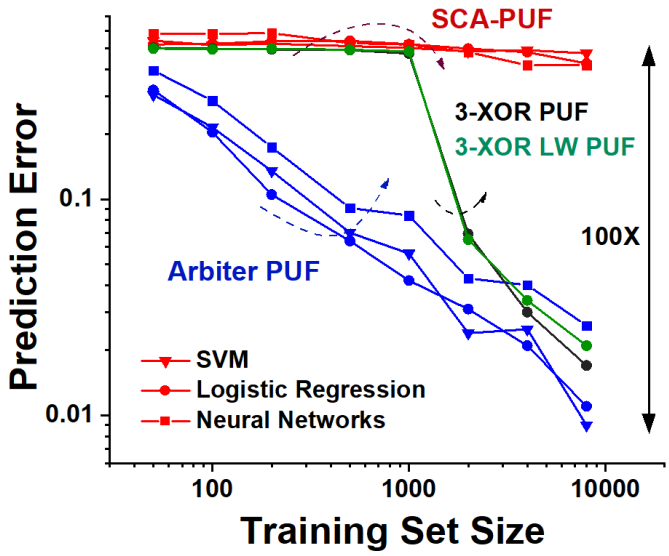
Figure 2.14: Inter-die and intra-die Hamming distance distributions.

of how different two arbitrary PUF instances will be. The ideal inter-die HD, when normalized to the total number of output bits, is 0.5. The measured normalized inter-die HD of the SCA-PUF has average of $\mu = 0.499$ and the standard deviation of $\sigma = 0.043$, the distribution is shown in Figure 2.14.

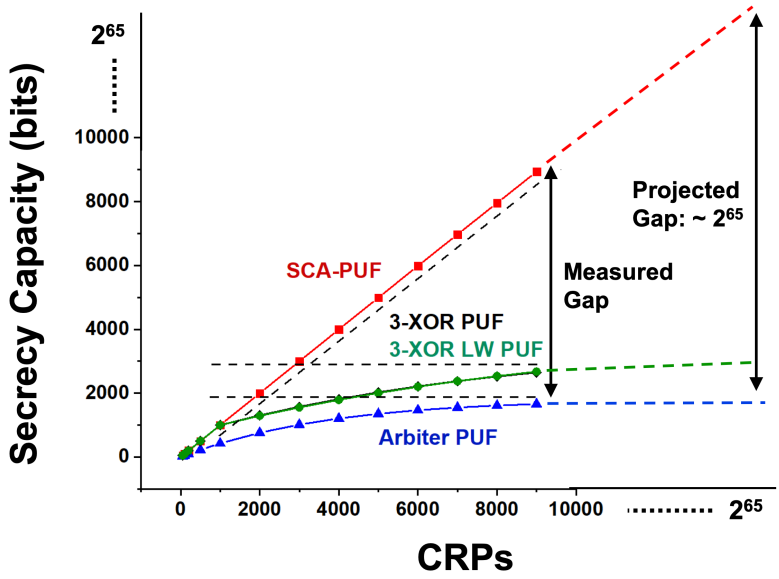
Further, for an ideal PUF, the fraction of "1"s and "0"s among the response bits should be equal. This measure of uniformity is defined as Hamming weight (HW) of the PUF output string. The randomness metric also quantifies uniformity but in min-entropy form. The ideal uniformity and randomness are both 0.5. The measured average uniformity is $\mu = 0.528$ ($\sigma = 0.109$), and the average randomness is $\mu = 0.528$ ($\sigma = 0.062$).

2.3.4 Resilience to ML attacks

An adversary may attempt to overcome the authentication guarantees offered by a PUF using machine learning attacks. The PUF challenge-



(a)



(b)

Figure 2.15: Machine learning attacks on the 65-bit SCA-PUF and arbiter PUF: (a) prediction error and (b) secrecy capacity results.

response behavior results from an underlying function with a limited number of unknown parameters, so this behavior can be learned by machine learning through observing a small set of CRPs (i.e. training set). The ability of a PUF to withstand machine learning attacks is a critical measure of strong PUF's security. Here, a suit of standard machine learning attacks is applied, including support vector machines, logistic regression, and neural networks. We ran these attacks on our SCA-PUF with the collected CRP dataset. We also ran the same attacks on 65-bit arbiter PUF, 3-XOR PUF [73], and 3-XOR lightweight (LW) PUF [73] for comparison. As shown in Figure 2.15a, the prediction error for a 65-bit SCA-PUF stays over 40% for the training set of 10^4 samples. This is ~100× higher than for the 65-bit arbiter PUF, 3-XOR PUF, and 3-XOR LW PUF.

ML-attack vulnerability affects the usability of a PUF in a Secure Key Generation (SKG) algorithm [34]. In such a setting, PUF responses serve as source of entropy from which secret keys can be extracted. The secrecy capacity $S(X)$ of a fuzzy secret X is defined as the theoretical maximum number of secure key bits that can be extracted from X. For a vector of N PUF response bits $X^N = (X_1, X_2, ..., X_N)$ with X_i as a single response bit, the secrecy capacity can be upper bounded as $S(X^N) \leq \sum_{i=1}^{N} h(SR(i-1)) - N \cdot h(p_e)$. Here $SR(i-1)$ is the prediction success rate of machine learning attack after observing $(i-1)$ CRPs, p_e is the average BER of PUF responses. $h(p)$ is the binary entropy function: $h(p) \equiv -p \log_2 p - (1-p) \log_2 (1-p)$. Figure 2.15b shows that the secrecy capacity increases linearly with the number N of CRPs for SCA-PUF while it saturates for the arbiter PUF, 3-XOR PUF, and 3-XOR LW PUF. This indicates that SCA-PUF has a much higher secrecy capacity (~2^{65} gap between SCA and the other three PUFs).

2.3.5 Area, Operating Frequency, and Power Consumption

As shown in Figure 2.9, the entire SCA-PUF design, including current mirrors, SCAs, the CMFB circuit and the comparator, occupies $44700\mu m^2$. Each single SCA occupies $6240\mu m^2$. Notice that when the number of challenge bits increases, only the SCA area increases.

The operating frequency is limited by the settling time of the SCA output voltage. The average frequency across 50 SCA-PUFs is measured to be 6Kb/s at $V_{DD} = 1.2V$. The average power consumption and energy/bit are 68nW and 11pJ/bit, respectively.

2.3.6 Comparison with State of the Art

In Table 2.1, the proposed SCA-PUF is compared with state-of-the-art strong PUFs, including delay PUFs [92, 54, 50], ring orsillator (RO) PUFs [76], and SRAM-based PUFs [37]. The most important advantage of SCA-PUF is the resilience to machine learning attacks and large secrecy capacity. Previous arbiter delay PUFs have been shown to be vulnerable to machine learning attacks and thus have quite small secrecy capacity. The BER obtained after the calibration-based CRP selection is also much lower than in previous works, which reduces post-processing cost of error correction. The uniqueness and uniformity of SCA-PUF are excellent compared to previous work. The power consumption is lower than previous work. The operating range of SCA-PUF is wide enough to tolerate temperature and supply variations.

Table 2.1: State-of-the-art strong designs.

	This work	[37]	[92]	[54]	[50]	[76]
Technology	0.13μm	28nm	40nm	65nm	0.18μm	90nm (FPGA)
Type	SCA PUF	SRAM-based	Delay PUF			RO PUF
Post-attack security (prediction error on 10^4 CRPs)	40%	10%	-	1%		1% [73]
Secrecy capacity (bits) (on 2^{65} CRPs)	$\sim 2^{65}$	-	-	~2000		-
Number of CRPs	$\sim 3.7\times10^{19}$	1.17×10^{11}	$\sim 5.5\times10^{28}$	$\sim 1.8\times10^{19}$	1.4×10^{20}	523776
BER in the worst case	9% (averaged) 0.4% (42% CRPs discarded)	3.17%	9%	4.5%	4.8%	0.48%
Uniqueness (mean inter-die HD)	0.499	0.481-0.495	0.501	-	0.400	0.462
Uniformity (mean HW)	0.528	-	0.472	-	-	-
Power (μW) / Energy (pJ/bit)	0.068 / 11.0	-/0.097	28.4 / 17.75	-	-	-
Supply range	1.08-1.32V	0.5-0.9V	0.7-1.2V	1.2-1.44V	$\pm$2%	1.08-1.2V
Temperature range	-20-80°C	0-80°C	-25-125°C	-40-85°C	27-67°C	20-120°C

Chapter 3

Fresh Re-Keying with Strong PUFs: a New Approach to Side-Channel Security

3.1 Introduction

[1] Side-channel attacks are a crucial threat for cryptographic schemes running on embedded devices. The adversary can recover secret information by analyzing side-channel analog behavior of a device, such as power consumption, during the execution of a cryptographic algorithm [47]. Traditional countermeasures against power-based side-channel attacks include masking and hiding. Masking adds randomness to key-dependent computations to remove correlations between the key and the intermediate values [30], and hiding adds redundant activity into the circuit to achieve constant power consumption [83]. Although these methods improve the security of the system by increasing the number of traces required to extract the key, they do not fully eliminate the side-channel threat [94, 87].

3.1.1 Basic Fresh Re-keying Scheme

To remove above side-channel vulnerabilities, Medwed *et al.* proposed a new scheme called *fresh re-keying* that regularly updates the key to limit per-key side-channel exposure [62]. Figure 3.1a shows the fresh re-keying scheme

[1] Contents of this chapter are based on [88] in which the author made substantial contributions to the development and design of the main idea.

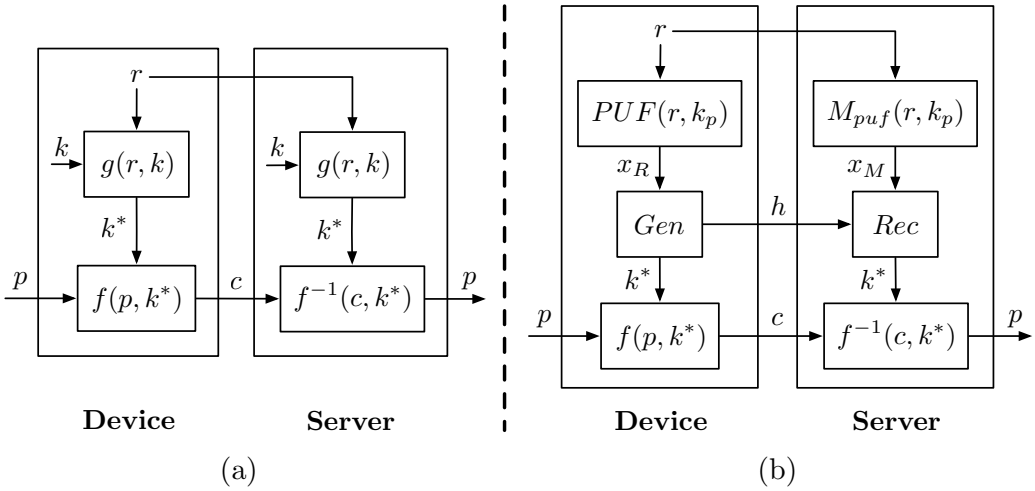


Figure 3.1: (a) The original re-keying scheme; (b) Our proposal: re-keying with a strong PUF.

used in a symmetric-key encryption scenario between a device and a server. We are specifically interested in applications where the device is deployed in the field and hence is the target of physical side-channel attacks while the sever operates in a physically-secured location. Internet-of-Things applications are a typical example of this scenario where a leaf-node device needs to send data to a cloud server. The system needs to transmit information confidentially from the device to the sever by encrypting it with a key. The device and the server rely on a pre-shared master key k to achieve this goal. Both the device and the server contain two blocks: the *key update function* g and the encryption function f (e.g., the AES block cipher). The key update function g generates a fresh key k^* using the master key k and a public nonce r. The encryption function f uses the fresh key k^* to encrypt a plaintext message p into a ciphertext output c. The device then sends r and c to the server. The server uses this information first to compute the fresh key and then to decrypt

the ciphtertext.

Unfortunately, fresh re-keying does not eliminate the problem of side-channel vulnerability but rather defers it to another block. Differential Power Analysis (DPA) on function f becomes impractical because, in the worst case, every plaintext can be encrypted with a different k^*. Therefore, f only needs to be secure against Simple Power Analysis (SPA), which is relatively easier to achieve. The adversary, however, can still apply DPA on function g to extract the master key k and thus obtain all fresh secret keys. Hence, g needs to be carefully designed to be secure against both SPA and DPA attacks. Unfortunately, several re-keying constructions were shown to be vulnerable to such side-channel attacks [67, 84, 20]. These vulnerabilities motivate our search for a fundamentally different approach to designing the key update function g.

3.1.2 New Approach: Re-keying with Strong PUFs

In this project, we propose to replace the algorithmic (mathematical) key update function g with a physical object, namely, the PUF, in a way that the construction becomes more secure against the side-channel vulnerabilities of earlier designs. The desired properties are achieved by adopting a strong PUF, that is, a PUF capable of producing an exponentially large number of challenge-response pairs (CRPs). For our purposes, such a strong PUF can be abstracted as a hardware hash function with a random key, k_p, defined by the unique physical instantiation of process variation values (e.g., gate delays or threshold voltages).

Figure 3.1b shows the operation of the proposed fresh re-keying scheme. The strong PUF accepts an input challenge r and generates a response x_R. A

reverse fuzzy extractor (RFE), consisting of two functions, Gen and Rec, is used to produce a high-entropy key along with helper data [21, 85]. Specifically, Gen receives the PUF response x_R and generates a key k^* and helper data h. The server generates the enrolled response x_M based on a pre-stored model of the PUF (M_{puf}). The response x_M is used by Rec, along with helper data h, to recover k^*.

We now analyze the requirements for ensuring the higher security of PUF-based re-keying. Algorithmic re-keying using a digital implementation of g is vulnerable because observing r and power of g allows an attacker to recover the master key k. Therefore, PUF-based re-keying needs to eliminate this vulnerability, such that observing r and the power of the PUF does not reveal the key k_p. However, Becker *et al.* showed that it is possible to derive k_p, for some PUFs, by utilizing power side-channel information [8]. We first hypothesize that a PUF that is constructed to be intrinsically resilient against a CRP-only modeling attack is also resilient against machine learning (ML) attacks utilizing side-channel information. Indeed, if the side-channel does not reveal any information beyond what is already contained in CRPs, then it cannot supply any additional information to the learning algorithm, and thus, a side-channel modeling attack is also likely to fail. With this objective in mind, we investigate the use of the SCA-PUF which is introduced in Section 2 and is explicitly constructed to be resilient to CRP-based attacks. As our experiments show, it remains secure against the attack by Becker *et al.* [8].

Because it is undesirable for the server to store PUF CRPs directly, our scheme relies on a model of the PUF. The server performs a one-time extraction of k_p during enrollment and uses it later to generate fresh keys k^*. The extraction is made possible via a one-time interface, such as a fuse that

is disabled after enrollment, so that the extraction cannot be done after the device is deployed [71, 14].

3.2 Background

3.2.1 Previous Work on Re-Keying

Several proposals for designing the key update functions have appeared [1, 66, 23, 80, 78]. Abdalla *et al.* [1] proposed a re-keying scheme which uses pseudo-random functions (PRFs) in a tree-based approach where the root node is the master key. Since the standard HMACs are not suitable for re-keying [24] [79], Pereira *et al.* [66] developed a new construction based on a leakage-resilient MAC. Dziembowski *et al.* [23] formulated a fresh re-keying method with hard learning problems and gave two constructions based on Learning Parity with Leakage and Learning with Rounding. Most recently, Taha *et al.* [80] showed a fresh re-keying scheme using a non-linear feedback shift register (NLFSR) to generate a stream of fresh keys based on a secret seed. It was later enhanced via the use of a stateless function for improved side-channel resistance [78].

Unfortunately, several re-keying constructions were shown to be vulnerable to side-channel attacks [67, 20, 84]. Pessl *et al.* [67] used side-channel information by casting the key recovery problem as Learning Parity with Noise (LPN) problem. Unterluggauer *et al.* [84] showed that the frequent re-keying in leakage-resilient streaming modes can cause constant plaintexts to be vulnerable to DPA. The NLFSR-based approach of Taha *et al.* [80] was also broken by DPA attacks [20].

3.3 Side-Channel Security Analysis of PUF-based Re-Keying

To achieve our security objectives — of being secure against the attack by Becker *et al.*, — we propose to use a strong PUF explicitly designed to be secure against CRP-only modeling attacks. The SCA-PUF was introduced by Kalyanaraman *et al.* [39] and further developed by Xi *et al.* [89] who demonstrated an ASIC implementation in a 130nm CMOS process. In this section, we first analyze the power consumption of SCA-PUF and show that it is infeasible for the attacker to read out PUF responses directly from power traces. We then analyze the security of SCA-PUF against power side-channel attack by Becker *et al.* [8]. We also analyze the security of other components in the system against SPA attacks.

3.3.1 Security of SCA-PUF against Simple Power Analysis

We first evaluate information leakage of a standalone SCA-PUF via the power side-channel within the framework of SPA. We need to ensure that it is infeasible for the adversary to directly read out a PUF response from the power traces obtained during a PUF evaluation. SCA-PUF has a largely data-independent power consumption: the currents through the two PUF arrays and the control block CMFB are constant, and the comparator uses two identical registers to store complementary outputs. This makes power drawn in generating a 1 and a 0 nearly identical.

We evaluate the possibility of a direct read-out via a statistical experiment. The power traces are generated by a running SPICE simulation of the 65-bit SCA-PUF in 130nm. Fig 3.2 shows the distribution of 2000 power values of both 0- and 1-responses at the time point of maximum power dif-

ference. We evaluate the probability of an attacker successfully classifying a just-observed power sample given the shown distribution. In a scenario favorable to an adversary, we assume that the attacker has a complete view of the distribution, namely, that it knows whether each trace is due to 0/1. We show that the probability of a correct identification is close to random guess.

Let $\mathcal{N}_0$ represent the Gaussian distribution of power for response 0, $\mathcal{N}_1$ for response 1, and $\mathcal{T}$ for the joint distribution. Let x represent a power value and z_x the label of it (whether it is due to 0 or 1 output). We denote as $\tilde{z}_x$ the label decided by the adversary. According to the *maximum a-posteriori probability rule*, the following Bayesian classifier produces the smallest labeling error [22]:

$$\tilde{z}_x = \begin{cases} 0, & \text{if } \Pr[z_x = 0|x] > \Pr[z_x = 1|x] \\ 1, & \text{otherwise} \end{cases} \tag{3.1}$$

We use $\Pr[\mathcal{N}_0] = 0.472$ to represent the uniformity of the SCA-PUF. For a given x, the probability that $z_x = 0$ is

$$\begin{aligned} \Pr[z_x = 0|x] =& \frac{\Pr[z_x = 0, x]}{\Pr[x]]} = \frac{\Pr[\mathcal{N}_0, x]}{\Pr[x]} \\ =& \frac{\Pr[\mathcal{N}_0] \cdot \Pr[x|\mathcal{N}_0]}{\Pr[\mathcal{N}_0] \cdot \Pr[x|\mathcal{N}_0] + \Pr[\mathcal{N}_1] \cdot \Pr[x|\mathcal{N}_1]} \\ \Pr[z_x = 1|x] =& \frac{\Pr[\mathcal{N}_1] \cdot \Pr[x|\mathcal{N}_1]}{\Pr[\mathcal{N}_0] \cdot \Pr[x|\mathcal{N}_0] + \Pr[\mathcal{N}_1] \cdot \Pr[x|\mathcal{N}_1]}. \end{aligned} \tag{3.2}$$

The probability that the adversary correctly chooses the label of x is

$$\begin{aligned} p_{corr}(x) =& \Pr[\tilde{z}_x = 0|x] \cdot \Pr[z_x = 0|x] + \\ & \Pr[\tilde{z}_x = 1|x] \cdot \Pr[z_x = 1|x]. \end{aligned} \tag{3.3}$$

Using the above we calculate the joint distribution $\mathcal{T}$ and $p_{corr}(x)$, Fig 3.3. The expectation of $p_{corr}(x)$ is $\mathbb{E}[p_{corr}(x)] = 0.584$. We use the probability

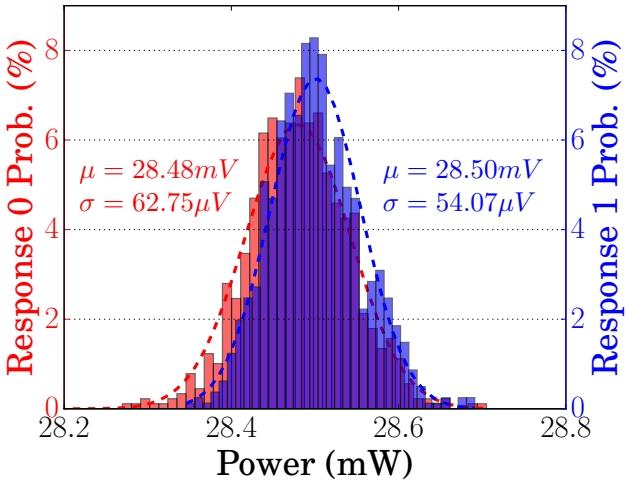


Figure 3.2: Power distribution of response 0 and 1.

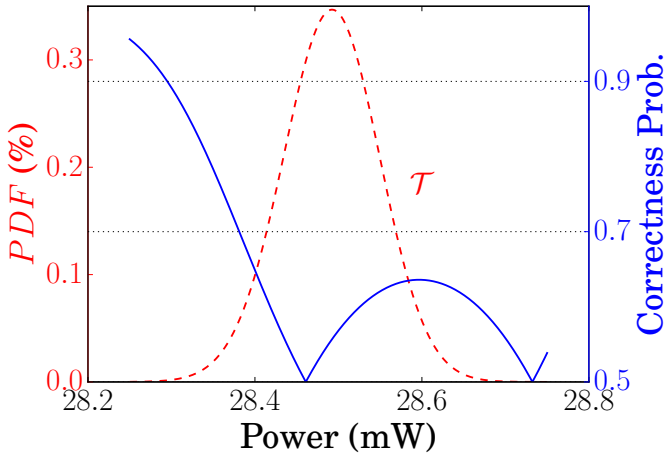


Figure 3.3: Joint distribution $\mathcal{T}$ and correctness probability of the attacker.

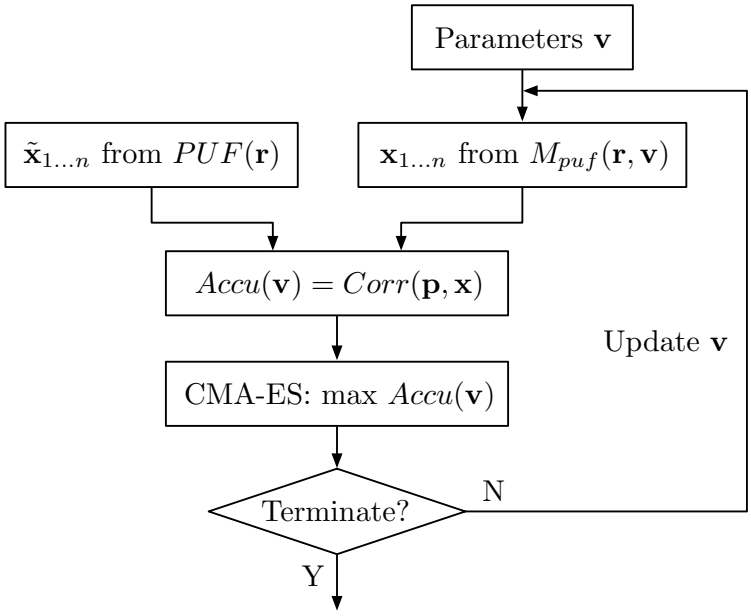
of a correct guess to bound the min-entropy of the SCA-PUF output for the purpose of subsequent fuzzy extractor construction. We use the bound on min-entropy of the PUF output Y proposed in [18]

$$\tilde{\mathbf{H}}_{\infty}(Y) \leq -\log_2(Accuracy(Y)) \tag{3.4}$$

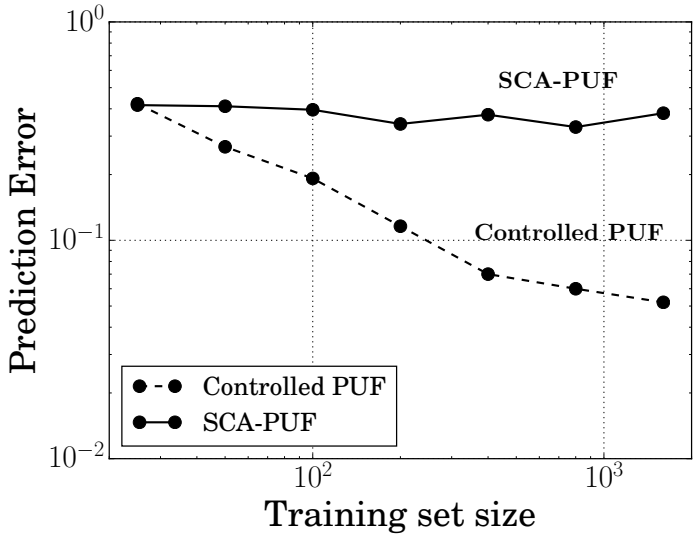
where $Accuracy(Y)$ is the probability of an accurate bit identification, which we estimate by $\mathbb{E}[p_{corr}(x)]$. With that, we compute the upper bound on min-entropy of the SCA-PUF response bit, subjected to the simple power analysis, to be 0.776.

3.3.2 Security of SCA-PUF against Power-Based Modeling Attack

Above, we established that it is infeasible for the attacker to directly read out PUF responses from individual power traces. However, an attack that uses power side-channel to first build a model of the PUF and then use it to generate outputs, has been described by Becker *et al.* [8]. This attack uses Covariance Matrix Adaptation Evolution Strategy (CMA-ES) and power measurements to extract the secret delay parameters (k_p) of an arbiter PUF [8]. The attack is presented in a controlled PUF setting where the arbiter PUF response is hashed with a cryptographic function. Although hashing disables modeling-based attacks on the hash output, the power side-channel enables an attack before the hashing stage, directly on arbiter PUF responses. The key idea is that an accurate model of the PUF generates (accurate) responses that exhibit high correlation with the power measured during response generation. The degree of correlation reflects the accuracy of the PUF model. The attack, therefore, starts from a random PUF model and eventually converges to an accurate one by checking the model response through power consumption and evolving towards the one yielding higher correlation.



(a)



(b)

Figure 3.4: (a) Power side-channel modeling attack based on Covariance Matrix Adaptation Evolution Strategy (CMA-ES); (b) Result of the power side-channel modeling attack on SCA-PUF with CMA-ES. Vulnerability of controlled PUF is shown as reference.

We first validate our experimental setup by applying the power side-channel attack by Becker *et al.* [8] on the arbiter PUF in a controlled PUF setting and then test the security of SCA-PUF using the same methodology. The responses of SCA-PUF are generated via a DC SPICE simulation of SCA-PUF. We assume that the adversary has access only to a generic SPICE device model for a certain generation of silicon technology, e.g., a Predictive Technology Model. To model the arbiter PUF, we use a linear function of mapped inputs [59]. Because of the exceedingly high computational cost of doing SPICE simulations of a large PUF on a scale required by an evolutionary algorithm, we perform the experiment on a 15-bit version of controlled PUF and SCA-PUF.

Figure 3.4a shows the flow of the side-channel attack on a PUF with measured power traces provided to the adversary. Here we denote as $\mathbf{v}$ the random parameters in the PUF model $M_{puf}(\mathbf{v})$. We calculate the set of correlation values between the measured power traces of a PUF instance $\mathbf{p}$ and the corresponding responses $\mathbf{x}$ from the PUF model $M_{puf}(\mathbf{r}, \mathbf{v})$, and select the maximum correlation value as the best measure of correlation between them. We denote this correlation as $Corr(\mathbf{p}, \mathbf{x})$ and use it as the measure of model accuracy $Accu(\mathbf{v})$. The parameters $\mathbf{v}$ are updated on each iteration of CMA-ES. The target is to maximize the correlation $Corr(\mathbf{p}, \mathbf{x})$ and thus obtain the optimal $\mathbf{v}$ (i.e. the gate delays in the arbiter PUF or the V_{th} values in the SCA-PUF).

We use the `cma 2.3.1` library [31] to implement the CMA-ES algorithm. We use a growing number of CRPs for training the model and use 1000 CRPs for testing. Figure 3.4b shows the result of CMA-ES attack on arbiter PUF and SCA-PUF. The attack can predict the responses of the controlled

arbiter PUF successfully while failing to build an accurate model of the SCA-PUF. From Figure 3.4b we see that the residual error probability after 1600 power traces is $p_{err} = 0.382$ for SCA-PUF. For the controlled arbiter PUF, the error is about 0.05. We repeat the calculation of min-entropy of the SCA-PUF output Y, based on the definition by Delvaux *et al.* [18], using the residual error of the model that predicts a PUF response:

$$\tilde{\mathbf{H}}_{\infty}(Y) \leq -\log_2(Accuracy(model)) \tag{3.5}$$

where $Accuracy(model)$ is the accuracy of the model produced by the CMA-ES above at the end of the training phase. With that, we compute the upper bound on min-entropy of the SCA-PUF response bit, subjected to a modeling attack with ML, to be $-\log_2(1 - p_{err}) = 0.694$.

3.3.3 Security of Other Components against Power Side-Channel Attacks

In any re-keying scheme, while the key update function should be secure against side-channel attacks with multiple measurements, other components should be secure against SPA attacks. Therefore, in addition to SCA-PUF, other modules inside our system such as the fuzzy extractor [41] can also be targeted with SPA-style attacks. Since this problem is orthogonal to the security requirements of the PUF core, we do not address them in this paper. Generic countermeasures to prevent single measurement attacks include shuffling or randomizing the data flow [82].

3.4 Enrollment Modeling of SCA-PUF

In this section, we develop an efficient method to extract the model of SCA-PUF during the enrollment phase and test its robustness on the fabri-

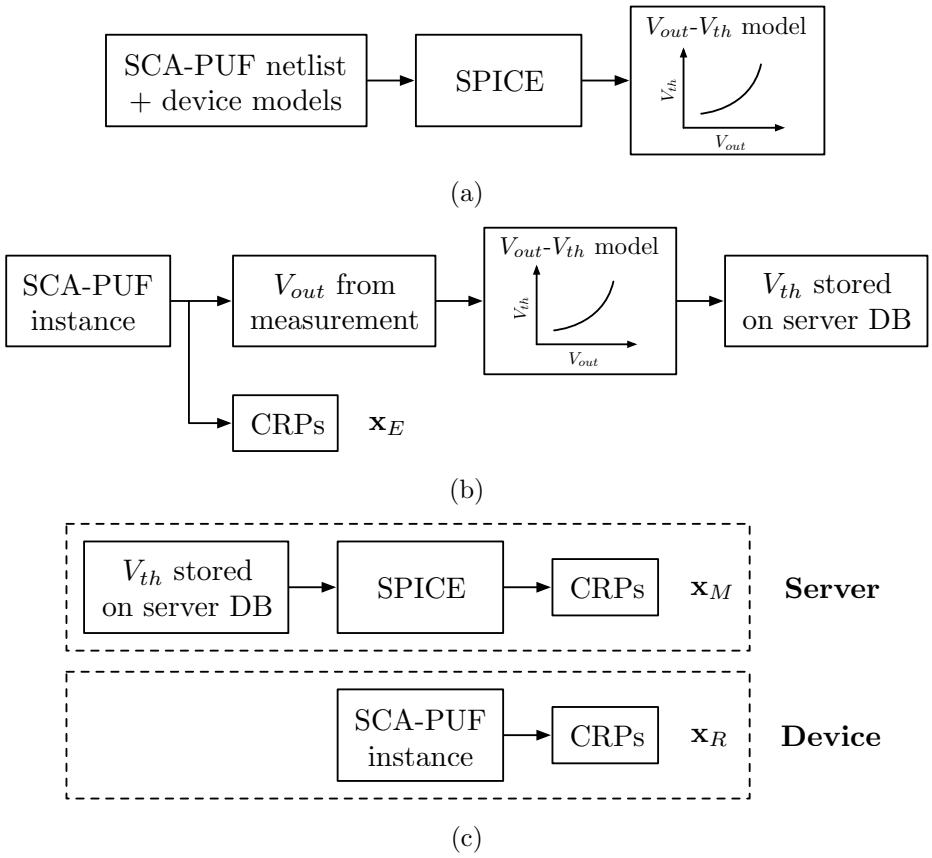


Figure 3.5: (a) Step I: V_{out}-V_{th} model extraction; (b) Step II: stochastic transistor V_{th} extraction (enrollment); (c) Step III: CRPs generation on both sides.

cated SCA-PUF instances. Since the server needs to enroll each unique PUF instance, the model of the PUF should be easy to store. The behavior of SCA-PUF is pre-dominantly impacted by the variations of the threshold voltages of a subset of transistors (the so-called stochastic transistors). Thus, the modeling challenge reduces to building a procedure for extracting and storing V_{th} values of SCA-PUF.

3.4.1 Efficient PUF Enrollment

Step I: Population-Based PUF Pre-Characterization: The premise of the end-to-end encryption is that at enrollment time, privileged access to the PUF, that allows easy characterization and model-building, exists but is disabled after enrollment to disallow such modeling in the field. The SCA-PUF behavior is uniquely characterized by a set of threshold voltages. However, measuring them directly is not possible. We developed a flow based on measuring the analog output of the PUF and deducing the value of the threshold voltages from the pre-characterized population-based models.

In Step I, the goal is to extract a population-based model relating threshold voltages of each transistor in the SCA array to the analog output voltage. That set of relations is captured in the form of a look up table. The model is extracted via a circuit (SPICE) simulation of the SCA-PUF circuit using the true device models of a manufacturer. Figure 3.5a illustrates the characterization flow. Circuit simulation is used to get V_{out} vs. V_{th} for each stochastic transistor. During extraction, only one stochastic transistor at a time is connected into the SCA network by setting the challenge bit controlling this transistor to be 1. Figure 3.6 shows the V_{out}-V_{th} model for a 65-bit SCA-PUF with a 5×13 array.

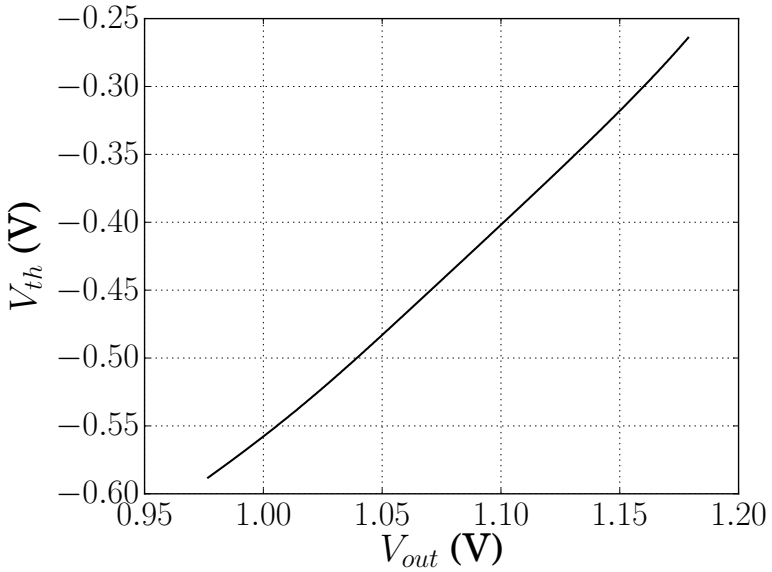


Figure 3.6: The enrollment uses a pre-characterized relation between measured V_{out} and inferred V_{th}.

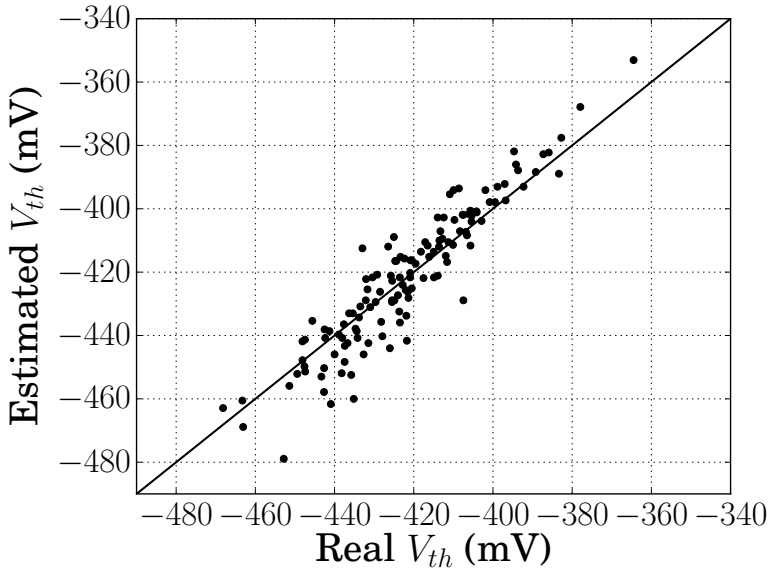


Figure 3.7: Accuracy of V_{th} prediction based on SCA-PUF model.

Step II: Per-Device V_{th} Extraction and Modeling: To enroll a unique PUF device on the server side, we need to extract the V_{th} values of each stochastic transistor in the SCA network. This is realized by measuring V_{out_a} and V_{out_b} for a unique input and using the V_{out}-V_{th} model to determine the corresponding V_{th}. After enrollment, the direct accesses to V_{out_a} and V_{out_b} are disabled (by burning a fuse) preventing the adversary from repeating the process in the future. Figure 3.5b illustrates Step II. As in Step I, for each measurement, only one stochastic transistor is connected within the SCA network. For a t-bit input SCA-PUF ($t = mn$ in Section 3.2), the procedure is carried out for all the $2t$ stochastic transistors. The $2t$ V_{th} values are then stored in the server database (DB). We define the responses generated by SCA-PUF at the enrollment step as $\mathbf{x}_E$.

Step III: Model-Based CRP Generation: The server regenerates CRPs using the V_{th} values stored in DB (Figure 3.5c). Importantly, accurately predicting CRPs requires running a circuit (SPICE) simulation and our assumption is that the server-side authenticator is equipped with the PUF netlist, the SPICE models, and a circuit simulator. The server only runs DC simulation to get the PUF responses. Since the server can run simulations in a distributed manner, the simulation time can be made small. We denote the responses generated by the SCA-PUF model as $\mathbf{x}_M$, and evaluate the accuracy of the model by comparing the modeled response (from the server) with the PUF response values directly produced by the proving (client) device. We denote the device response as $\mathbf{x}_R$. Note that because the responses of SCA-PUF on the device are produced in the field, they may be impacted by temperature and supply voltage variation, and thus $\mathbf{x}_R$ is, in general, different from $\mathbf{x}_E$.

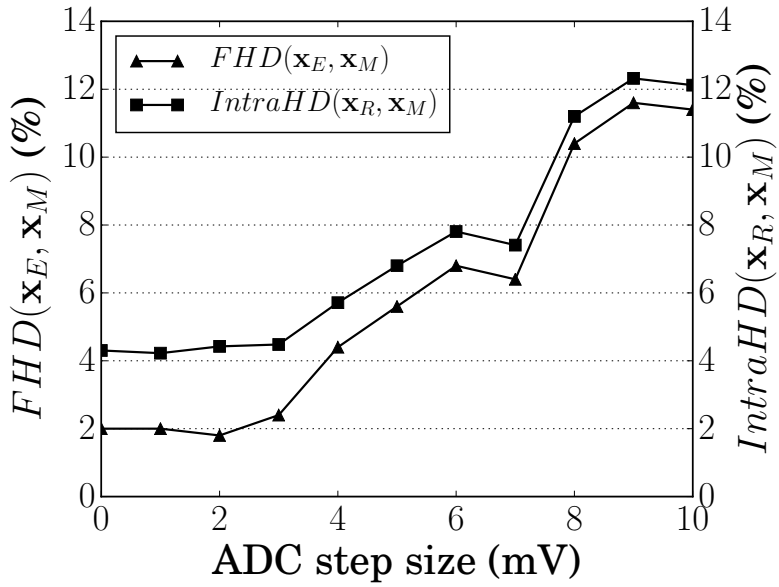


Figure 3.8: $FHD(\mathbf{x}_E, \mathbf{x}_M)$ and $IntraHD(\mathbf{x}_R, \mathbf{x}_M)$ for several ADC resolutions.

3.4.2 Evaluation of SCA-PUF Enrollment Modeling

First, we evaluate the accuracy of the procedure to estimate V_{th} values. The Monte Carlo simulation of the SCA-PUF circuit obtains the V_{th} values of the stochastic transistors. The extracted LUT and the measured V_{out} are then used to estimate V_{th}. Figure 3.7 plots the estimated V_{th} vs. the actual V_{th}. The root min square error (RMSE) is 3.5mV (roughly, 1%).

We then test the accuracy of the CRP prediction via a simulation study. We evaluate the fractional Hamming distance (FHD) between the enrolled PUF responses $\mathbf{x}_E$ and the responses generated by the model ($\mathbf{x}_M$) at the nominal conditions (25°C and 1.2V V_{DD}) using 500 challenges. The value of $FHD(\mathbf{x}_E, \mathbf{x}_M)$ is 2%. We also evaluate the intra-class HD ($IntraHD$) between $\mathbf{x}_M$ and the reconstructed responses $\mathbf{x}_R$ across the temperature range from -

20°C to 80°C and ±10% V_{DD}. The mean value of $IntraHD(\mathbf{x}_R, \mathbf{x}_M)$ is 4.5%.

The resolution of measurement equipment is also a source of inaccuracy of the responses generated from the PUF model. We also evaluate the accuracy of the model with different levels of resolution of the measurement equipment. We denote the step size of an ADC as Δ. Since the supply voltage $V_{DD} = 1.2V$, the minimal voltage difference of an 8-bit ADC is $\Delta = \frac{V_{DD}}{2^8} = \frac{1.2}{2^8} \approx 4.7$mV. The result of $FHD(\mathbf{x}_E, \mathbf{x}_M)$ and $IntraHD(\mathbf{x}_R, \mathbf{x}_M)$ vs. Δ from $0mV$ to $10mV$ is shown in Figure 3.8.

We further validate the practicality of the proposed flow using the silicon SCA-PUF chip fabricated in 130nm. The 8-bit ADC has a resolution of 4.7mV. According to Figure 3.8, we expect $FHD(\mathbf{x}_E, \mathbf{x}_M)$ to be 5%-6%, and $IntraHD(\mathbf{x}_R, \mathbf{x}_M)$ to be 6%-7%. The silicon test shows that $FHD(\mathbf{x}_E, \mathbf{x}_M)$ is 6.0% and $IntraHD(\mathbf{x}_R, \mathbf{x}_M)$ is 7.9%. Thus, silicon results are consistent with simulation predictions. We also measured the intra-class HD between the enrolled PUF responses $\mathbf{x}_E$ and the reconstructed responses $\mathbf{x}_R$ across the temperature range from -20°C to 80°C and ±10% V_{DD}: the mean of $IntraHD(\mathbf{x}_R, \mathbf{x}_E)$ is 5.1%. While $IntraHD(\mathbf{x}_R, \mathbf{x}_M)$ is larger than $IntraHD(\mathbf{x}_R, \mathbf{x}_E)$ due to the inaccuracy of $\mathbf{x}_M$, this difference is acceptable because of error-correction embedded in the reverse fuzzy extractor.

3.5 PUF-based Re-Keying System Design

In this section, we discuss the design of an end-to-end encryption system using the proposed re-keying scheme. We also discuss implementation details of the system.

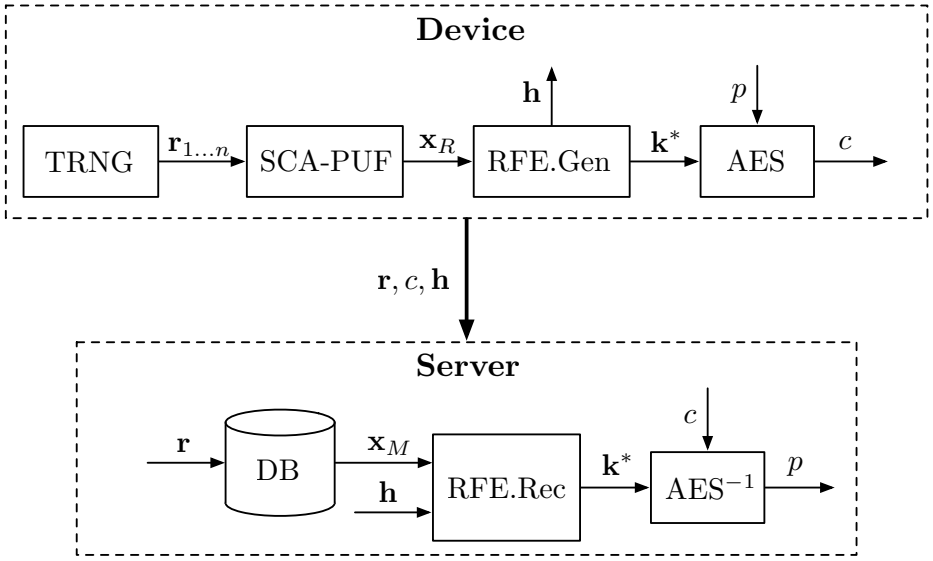


Figure 3.9: Building blocks of the PUF-based re-keying scheme.

	Device		Server
Enroll.$(1\times)$	$\mathbf{V}_{out1\ldots2t} \leftarrow \text{PUF}$	$\xrightarrow{\mathbf{V}_{out}}$	$\mathbf{V}_{th1\ldots2t} \leftarrow \text{LUT}(\mathbf{V}_{out})$
			$\text{DB}[\text{M}_{\text{puf}}, \mathbf{V}_{th}]$
Encrypt.(∞)	$\mathbf{r}_{1\ldots n} \leftarrow \text{TRNG}()$	$\xrightarrow{\mathbf{r}}$	$\mathbf{x}_M \leftarrow \text{M}_{\text{puf}}(\mathbf{r}, \mathbf{V}_{th})$
	$\mathbf{x}_R \leftarrow \text{PUF}(\mathbf{r})$		
	$\mathbf{k}^* \leftarrow \text{TRNG}()$		
	$\mathbf{s} \leftarrow \text{BCH.Enc}(\mathbf{k}^*)$		
	$\mathbf{h} \leftarrow \mathbf{x}_R \oplus \mathbf{s}$	$\xrightarrow{\mathbf{h}}$	$\tilde{\mathbf{s}} \leftarrow \mathbf{x}_M \oplus \mathbf{h}$
			$\mathbf{k}^* \leftarrow \text{BCH.Dec}(\tilde{\mathbf{s}})$
	$c \leftarrow \text{AES}(p, \mathbf{k}^*)$	$\xrightarrow{c}$	$p \leftarrow \text{AES}^{-1}(c, \mathbf{k}^*)$

Figure 3.10: End-to-end encryption system with the PUF-based re-keying scheme.

3.5.1 End-to-End Encryption

Fuzzy extractors are a common method to address PUF noise and to generate stable entropy from noisy PUF responses [21]. To reduce device-side complexity of the system, we use a reverse fuzzy extractor (RFE) that moves the complex error correction process from the device to the server [85].

Figure 3.9 shows the building blocks of the proposed re-keying scheme. The device contains an SCA-PUF, a true random number generator (`TRNG`), an RFE generation block (`RFE.Gen`) and an encryption function (e.g. Advanced Encryption Standard (`AES`)). The server has a database (`DB`) to store PUF enrollment data, an RFE reconstruction block (`RFE.Rec`), and the decryption function to recover the plaintext message.

Our threat model assumes that the `DB` storing V_{th} values and the PUF model is secret. We assume that the device operates in the field and hence is the target of power side-channel attacks. By contrast, we consider server to be in a physically secure location so our threat model does not include side-channel attacks on the server. The adversary can eavesdrop on $\mathbf{r}$, c, and $\mathbf{h}$. The RFE construction ensures that there is sufficient left-over entropy on the PUF response (using the pessimistic $(n-k)$ bounds) so disclosing helper data $\mathbf{h}$ is secure even against state-of-the-art attacks [19].

Figure 3.10 lists the operations of the proposed end-to-end encryption method with PUF-based re-keying. During the enrollment phase, the server measures V_{out} values through a one-time interface, extracts V_{th} value of each stochastic transistor from the V_{out}-V_{th} model, and stores them in the server `DB`. During each encryption, `TRNG` generates a challenge set $\mathbf{r}_{1...n}$ for the SCA-PUF, where $\mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_n$ are the t-bit challenges. The SCA-PUF generates noisy responses $\mathbf{x}_R$. The error correction follows the code-offset construction

Table 3.1: Required BCH code parameters and PUF response bits.

Profile	×BCH	n_{BCH}	k_{BCH}	t_{BCH}	$\log_{10} EER$	Entropy	n_{PUF}
A1	1	2047	760	153	-6.23	134.28	2047
A2	1	4095	1496	293	-9.01	244.25	4095
A3	4	4095	1328	313	-12.14	304.98	16380

of RFE with BCH coding. The secret key $\mathbf{k}^*$ is a nonce, which is encoded into a codeword $\mathbf{s}$, and then XORed with the PUF responses to compute the helper data $\mathbf{h}$. The server then reverses these steps using the modeled PUF responses $\mathbf{x}_M$. If the Hamming distance between $\mathbf{x}_M$ and $\mathbf{x}_R$ is within the error-decoding capability of the BCH error decoder, the server reconstructs the correct $\mathbf{k}^*$. Finally, the server uses the decryption function with $\mathbf{k}^*$ to turn the ciphertext c into the correct plaintext p.

3.5.2 System Design: Details

Based on the analysis in section 3.3.1 and 3.3.2, we use the more conservative bound on min-entropy of the SCA-PUF response available to us. Thus, min-entropy of the PUF outputs in our design is, at most, 0.694. Given 5% BER of the SCA-PUF response, Table 3.1 provides a set of BCH parameters to meet the desired false rejection rate (FRR) and the overall entropy for three application profiles [53]: A1–128-bit key with 10^{-6} equal error rate (EER) [53], A2–128-bit key with 10^{-9} EER, and A3–256-bit key with 10^{-12} EER. n_{PUF} in Table 3.1 is the source response bit number of the PUF. In all cases, the SCA-PUF can generate a stable key for the desired entropy and reliability by using a proper BCH code.

The proposed system uses an SCA-PUF that has a CRP space of 2^{65}

bits, which may not be considered large enough for cryptographic purposes. Our main motivation in selecting this PUF is the validation of enrollment modeling (hence scalability) on the fabricated SCA-PUF instances. The methods we provide in this work can be extended to SCA-PUF circuits having a significantly larger CRP set.

Chapter 4

Implementation of A Lightweight Strong PUF Provably Secure to Machine Learning Attacks

4.1 Introduction

As discussed in Section 2, in order for a strong PUF to be an effective security primitive, the associated CRPs need to be unpredictable: given a certain set of known CRPs, it should be hard to predict the unobserved CRPs with high probability. In other words, strong PUFs are required to be resilient to modeling attacks. However, the effectiveness of modeling attacks via machine learning (ML) on many strong PUFs has been widely demonstrated [35, 72]. Most proposed modifications of the original arbiter PUF aimed to strengthen ML resistance, including the XOR arbiter PUF and the feed-forward PUF [51, 60, 77, 91], have also been broken via various ML attacks [7, 26, 27, 72]. By exploiting higher intrinsic nonlinearity, some strong PUFs [49, 89] exhibit empirically-demonstrated resistance to some ML algorithms. But empirical demonstrations of ML resistance are not fully satisfactory since they can never rule out the possibility of other more effective ML algorithms. No theoretical support for their ML resistance has been provided yet. The so-called controlled PUF setting [29] attempts to ensure the ML resistance via cryptographic primitives such as hash functions. However, the use of hash functions inside a PUF endangers the promise of a strong PUF as a lightweight structure. Strong PUF constructions using established cryptographic ciphers,

such as AES [11], have similar challenges. An additional challenge our paper addresses is that an ideal ML resistant PUF should exhibit security against both classical as well as quantum algorithms. In summary, the question of whether it is possible to engineer a provably ML secure lightweight strong PUF has been a long-lasting challenge [86].

In this project, we propose a strong PUF that is secure against ML attacks with both classical and quantum computers. The security is guaranteed by engineering a PUF for which modeling is, provably, a computationally hard problem. *The main insight is the mapping of ML attack resistance in a PUF to hardness of learning a decryption function of a cryptosystem.* As a formal framework, we adopt the probably approximately correct (PAC) theory of learning [64]. The specific insight, which allows us to build a novel strong PUF, is our reliance on the earlier proof that PAC-learning a decryption function of a semantically secure public-key cryptosystem entails breaking that cryptosystem [44, 45, 46]. The PAC non-learnability of a decryption function implies that with a polynomial number of samples, with high probability, *it is not possible to learn a function accurately by any means.* Specifically, we develop a PUF for which the task of modeling is equivalent to PAC-learning the decryption function of a learning-with-errors (LWE) public-key cryptosystem. LWE cryptosystems are based on the hardness of LWE problem that ultimately is reduced to the hardness of several problems on lattices [70]. The input-output mapping between the PUF and the underlying LWE cryptosystem can be briefly summarized as follows: challenge $\Longleftrightarrow$ ciphertext and response $\Longleftrightarrow$ decrypted plaintext. Notably, LWE is believed to be secure against both classical and quantum computers. Because of the intrinsic relation between the proposed PUF and the security of lattice cryptography we call our construction

the **lattice PUF**.

The lattice PUF is constructed using a POK, an LWE decryption function block, a linear-feedback shift register (LFSR), a self-incrementing counter, and a control block. The entire implementation is lightweight and fully digital. The LWE decryption function block is the core module of the lattice PUF, generating response (plaintext) to each submitted challenge (ciphertext). Design parameters of the LWE decryption function in the lattice PUF are chosen by balancing the implementation costs, statistical performance, and the concrete hardness of ML resistance. We develop a measure of ML security in terms of the total number of operations needed to learn the model of the PUF. Such concrete hardness is established by the analysis of state-of-the-art attacks on the LWE cryptosystem [52, 63] and evaluated by the estimator developed by Albrecht et al. [3]. Using this estimator, we say that a PUF has k-bit ML resistance if a successful ML attack requires 2^k operations. We implement the LWE decryption function that takes 1168-bit input challenges while guaranteeing 128-bit ML resistance.

The likely deployment model for strong PUFs presumes a clear distinction between a device being authenticated (a PUF proper) and an authenticating server sending challenges to the PUF and analyzing its responses. It is natural to assume that the server is less resource-constrained than the PUF. It is thus desirable to place the computationally-costly part of challenge generation, that utilizes a relatively more expensive encryption function of LWE, on the server.

The theoretical security guarantee, cited above, assumes that the inputs to the LWE decryption function are generated by an encryption function operating on the uniformly random plaintexts: we call such allowed queries

"challenges generated by a ciphertext distribution". However, we found that a direct implementation of the lattice PUF, in which the server fully generates ciphertext, is inefficient due to the well-known high ratio of ciphertext to plaintext: a PUF with a 128-bit concrete ML hardness requires transmitting $116.8K$ challenge bits in order to produce a 100-bit response string.

We solve this problem *by exploiting distributional relaxations allowed by recent work in space-efficient LWEs*. First, our approach replaces the component of ciphertext, dominating transmission cost, by a uniformly sampled random vector, such that the resulting distribution is statistically close to the original ciphertext distribution [2]. The advantage of the above replacement is that, as shown by Galbraith *et al.* [25], multiple simple pseudo-random number generators (PRNGs), including those based on a linear-feedback shift register (LFSR), are capable of producing it. Specifically, Galbraith *et al.* [25] shows that input challenges generated by PRNGs provide similar concrete security guarantees against standard attacks on LWE. The proposed strategy allows introducing a low-cost PRNG based on an LFSR and transmitting only a small seed. This results in a dramatic reduction of the effective challenge size. In the improved design with the same parameters chosen above, only 928 bits are needed to produce a 100-bit response. This is a 100X reduction of communication cost in authentication, in contrast to the direct PUF implementation as LWE decryption function.

The focus of the paper is a PUF that is secure against passive attacks in which the observed challenges can be used to derive an internal model of the PUF. However, the LWE decryption function is vulnerable to an active attack that supplies arbitrary input challenges. (As we show, this risk also carries into an LFSR-based variant). We overcome the risk of such an attack by adopting

the technique used by Yu *et al.* [93]: we introduce a self-incrementing counter to embed the counter value into a challenge seed. This makes the attack impossible as the counter restricts the attacker's ability to completely control input challenges to the LWE decryption function.

We implement the PUF on an FPGA which requires a 1160-bit secret key. The secret key is generated from POK bits. We use an FE with concatenated codes to reconstruct stable POK bits. The random source of the POK can be any weak PUF. The SRAM PUF power-up states are used in our paper. Assuming an average bit error rate (BER) of 5% for raw SRAM cells, the total number of raw SRAM bits needed is 6.5K, in order to achieve a key reconstruction failure rate of 10^{-6}. The LFSR utilizes a 256-bit seed. The self-incrementing counter produces a 128-bit output. Additional 128 bits are concatenated with the counter output to form the input seed to the LFSR. Thus, the resulting lattice PUF is able to achieve a CRP space of size 2^{136}. The mean BER (intra-class Hamming distance (HD)) is 4.43%. The lattice PUF also shows excellent uniformity and uniqueness. The hardware implementation on a Xilinx Spartan 6 FPGA utilizes only 45 slices for the lattice PUF logic and 233 slices for the concatenation-code-based FE. Compared to several known strong PUFs, the proposed PUF is significantly more resource-efficient.

4.2 LWE Decryption Functions Are Hard to Learn

This section formally defines ML resistance of strong PUFs via the notion of PAC learning and shows why LWE decryption functions are attractive for constructing post-quantum ML-resistant PUFs. In this section, we focus on passive attacks in which the attacker can observe the challenges sent to the verifier but is unable to generate challenges of his or her choice.

4.2.1 ML Resistance as Hardness of PAC Learning

A strong PUF can be modeled as a function $f : \mathcal{C} \to \mathcal{R}$ mapping from the challenge space $\mathcal{C}$ (usually $\{0,1\}^n$) to the response space $\mathcal{R}$ (usually $\{0,1\}$). We call f the true model of a strong PUF since it captures the exact challenge-response behavior.

ML attacks are usually performed by relying on a functional class of candidate models, collecting CRPs as the training data, and running a learning algorithm to obtain a model from the candidate class which best approximates the true model. In addition to the approximation quality, the criteria of evaluating the effectiveness and efficiency of the learning algorithm also include the sample and time complexity. To claim that a strong PUF is easy to learn, one can propose a learning algorithm which finds a CRP model with good approximation quality using a small number of sample CRPs and terminates in a short time. The converse is difficult: to claim that a PUF is hard to learn, one must show that all possible learning algorithms fail to provide models with good approximation quality, or they require a large number of CRPs or a long running time.

We argue that the only known framework for seeking a provable notion of ML resistance with a formal analysis of approximation quality, sample size, and time complexity is the PAC learning model [64]. We now formalize the passive modeling attack scenario in the context of PAC learning. A PAC-term for a true model f of a strong PUF is a concept. Denote as $\mathcal{F}$ the set of all possible PUF-realized functions (every instance of a PUF creates its unique functional mapping f). The set of candidate models used in the learning algorithm is the hypothesis set $\mathcal{H}$. The goal of a learning algorithm is to select a candidate model that matches the true model well. Importantly, as

shown later, the proof of PAC hardness guarantees that $\mathcal{H}$ does not have to be restricted to be the same as $\mathcal{F}$ of true models. This generalization permits a stronger *representation-independent* PAC-hardness proof. While not always possible, representation-independent hardness can be proven for PAC-learning of decryption functions ensuring that no matter how powerful and expressive the chosen $\mathcal{H}$ is, PAC learning decryption function requires exponential time.

Within the PAC model, CRPs in a training set are assumed to be independent and identically distributed (i.i.d.) under a certain distribution $\mathcal{D}$.

We say a set $\mathcal{F}$ of strong PUFs is PAC-learnable using $\mathcal{H}$, if there exists a polynomial-time algorithm $\mathcal{A}$ such that $\forall \epsilon > 0$, $\forall \delta > 0$, for any fixed CRP distribution $\mathcal{D}$, and $\forall f \in \mathcal{F}$, given a training set of size m, $\mathcal{A}$ produces a candidate model $h \in \mathcal{H}$ with probability of, at least, $1 - \delta$ such that

$$\Pr_{(\mathbf{c},r)\sim\mathcal{D}}[f(\mathbf{c}) \neq h(\mathbf{c})] < \epsilon.$$

In conclusion, our strategy is to say that a strong PUF is ML-resistant if it is not PAC-learnable (i.e., that it is PAC-hard). PAC-hardness implies that any successful ML attack requires at least an exponential running time.

4.2.2 Decryption Functions Are not PAC Learnable

What is critically important is that there exist functions that are known to be not PAC-learnable. Specifically, a class of decryption functions of secure public-key cryptosystems is not PAC-learnable, as established by Kearns *et al.* [44] and Klivans *et al.* [46]. We outline their proof below.

A public-key cryptosystem is a triple of probabilistic polynomial-time algorithms $(\text{Gen}, \text{Enc}, \text{Dec})$ such that: (1) Gen takes n as a security parameter

and outputs a pair of keys (pk, sk), the public and private keys respectively; (2) Enc takes as input the public key pk, encrypts a message (plaintext) r to return a ciphertext $\mathbf{c} = \text{Enc}(pk, r)$; (3) Dec takes as input the private key sk and a ciphertext $\mathbf{c}$ to decrypt a message $r = \text{Dec}(sk, \mathbf{c})$. We only need to discuss public-key cryptosystems encrypting 1-bit messages (0 and 1).

One of the security requirements of a public-key cryptosystem is that it is computationally infeasible for an adversary, knowing the public key, pk, and a ciphertext, $\mathbf{c}$ to recover the original message, r. This requirement can also be interpreted as the need for indistinguishability under the chosen plaintext attack (also often referred to as semantic security requirement) [43]. Given the encryption function Enc and the public key pk, the goal of an attacker is to devise a *distinguisher* $\mathcal{A}$ to distinguish between encryption $\text{Enc}(pk, r)$ of $r = 0$ and $r = 1$ with non-negligible probability:

$$|\Pr[\mathcal{A}(pk, \text{Enc}(pk, 0)) = 1] - \Pr[\mathcal{A}(pk, \text{Enc}(pk, 1)) = 1]| \geq \epsilon.$$

A cryptosystem is semantically secure if no polynomial-time attacker can correctly predict the message bit with non-negligible probability.

The connection between the above-stated security of a public-key cryptosystem and the hardness of learning a concept class associated with its decryption function was established by Kearns *et al.* [44] and Klivans *et al.* [46]. The insight is that PAC-learning is a natural result of the ease of encrypting messages with a public key. Since the encryption function Enc and the public-key pk is known, the distinguishing algorithm can sample independent training examples in the following way: (1) picking a plaintext bit r uniformly randomly from $\{0, 1\}$, (2) encrypting r to get the ciphertext $\mathbf{c} = \text{Enc}(pk, r)$. (We later refer to the resulting distribution of ciphertext as the "ciphertext

distribution".) Next, the distinguishing algorithm passes the set of training examples (($\mathbf{c}, r$)'s) into an algorithm for learning the decryption function $\mathrm{Dec}(sk, \cdot)$. The PAC learning algorithm returns a model $h(\cdot)$ that aims to approximate $\mathrm{Dec}(sk, \cdot)$. Using $h(\cdot)$, one could distinguish between ciphertexts stemming from $r = 0$ and $r = 1$ with non-negligible probability. This would entail violating the semantic security of the cryptosystem. Technically, this can be summarized as follows [44, 46].

Theorem 1. *If a public-key cryptosystem is secure against chosen plaintext attacks, then its decryption functions are not PAC-learnable (under the ciphertext input distribution).*

4.2.3 LWE Is Post-Quantum Secure

According to the cryptographic hardness above, decryption functions of any secure public-key cryptosystem, such as RivestShamirAdleman (RSA) and elliptic-curve cryptography (ECC), can be used to construct ML-resistant PUFs. However, integer-factoring-based cryptosystems, including RSA and ECC above, become insecure with the development of quantum computers. Among all post-quantum schemes [9], the LWE cryptosystem based on hard lattice problems appears to be most promising due to its implementation efficiency and stubborn intractability since 1980s.

A lattice $\mathcal{L}(\mathbf{V})$ in n dimensions is the set of all integral linear combinations of a given basis $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$ with $\mathbf{v}_i \in \mathbb{R}^n$:

$$\mathcal{L}(\mathbf{V}) = \{a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \ldots a_n\mathbf{v}_n : \forall a_i \in \mathbb{Z}\}.$$

The LWE problem is defined on the integer lattice $\mathcal{L}(\mathbf{V}) = \{(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle)\}$ with a basis $\mathbf{V} = (\mathbf{I}; \mathbf{s})$, in which $\mathbf{I}$ is an n-dimensional identity matrix and

$\mathbf{s}$ is a fixed row vector (also called the secret) in $\mathbb{Z}_q^n$. Throughout this paper, vectors and matrices are denoted with bold symbols with dimension on superscript, which can be dropped for convenience in case of no confusion. Unless otherwise specified, all arithmetic operations in the following discussion including additions and multiplications are performed in $\mathbb{Z}_q$, i.e. by modulo q.

For the lattice $\mathcal{L}(\mathbf{V}) = \{(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle)\}$ with dimension n, integer modulus q and a discrete Gaussian distribution $\bar{\Psi}_\alpha$ for noise, the LWE problem is defined as follows. The secret vector $\mathbf{s}$ is fixed by choosing its coordinates uniformly randomly from $\mathbb{Z}_q$. Next $\mathbf{a}_i$'s are generated uniformly from $\mathbb{Z}_q^n$. Together with the error terms e_i, we can compute $b_i = \langle \mathbf{a}, \mathbf{s} \rangle + e_i$. Distribution of $(\mathbf{a}_i, b_i)$'s over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ is called the LWE distribution $A_{\mathbf{s},\bar{\Psi}_\alpha}$. The most important property of $A_{\mathbf{s},\bar{\Psi}_\alpha}$ is captured in the following lemma:

Lemma 2. *Based on hardness assumptions of several lattice problems, the LWE distribution $A_{\mathbf{s},\bar{\Psi}_\alpha}$ of $(\mathbf{a}, b)$'s is indistinguishable from a uniform distribution in $\mathbb{Z}_q^n \times \mathbb{Z}_q$.*

Solving the decision version of LWE problem is to distinguish with a non-negligible advantage between samples from $A_{\mathbf{s},\bar{\Psi}_\alpha}$ and those generated uniformly from $\mathbb{Z}_q^n \times \mathbb{Z}_q$. This LWE problem is shown to be intractable to solve, without knowing the secret $\mathbf{s}$, based on the worst-case hardness of several lattice problems [70]. Errors e are generated from a discrete Gaussian distribution $\bar{\Psi}_\alpha$ on $\mathbb{Z}_q$ parameterized by $\alpha > 0$: sampling a continuous Gaussian random variable with mean 0 and standard deviation $\alpha q/\sqrt{2\pi}$ and rounding it to the nearest integer in modulo q. Notice that error terms are also essential for guaranteeing the indistinguishability: without noise $(\mathbf{a}, b)$ becomes deterministic and the secret $\mathbf{s}$ can be solved efficiently via Gaussian elimination methods.

We now describe a public-key cryptosystem based on the LWE problem above, developed by Regev *et al.* [70]:

Definition 3. *(LWE cryptosystem)*

- ***Private key:*** $\mathbf{s}$ *is uniformly random in* $\mathbb{Z}_q^n$.
- ***Public key:*** $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ *is uniformly random, and* $\mathbf{e} \in \mathbb{Z}_q^n$ *with each entry from* $\bar{\Psi}_\alpha$*. Public key is* $(\mathbf{A}, \mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e})$.
- ***Encryption:*** $\mathbf{x} \in \{0,1\}^m$ *is uniformly random. To encrypt a one-bit plaintext* r*, output ciphertext* $\mathbf{c} = (\mathbf{a}, b) = (\mathbf{A}^T\mathbf{x}, \mathbf{b}^T\mathbf{x} + r\,\lfloor q/2 \rfloor)$.
- ***Decryption:*** *Decrypt the ciphertext* $(\mathbf{a}, b)$ *to* 0 *if* $b - \langle \mathbf{a}, \mathbf{s} \rangle$ *is closer to* 0 *than to* $\lfloor q/2 \rfloor$ *modulo* q*, and to* 1 *otherwise.*

Notice that each row in the public-key $(\mathbf{A}, \mathbf{b})$ is an instance from the LWE distribution $A_{\mathbf{s},\bar{\Psi}_\alpha}$.

Correctness of the LWE cryptosystem can be easily verified: without the error terms, $b - \langle \mathbf{a}, s \rangle$ is either 0 or $\lfloor q/2 \rfloor$, depending on the encrypted bit. Semantic security of the LWE cryptosystem follows directly from the indistinguishability of the LWE distribution from the uniform distribution in $\mathbb{Z}_q^n \times \mathbb{Z}_q$. Ciphertexts $(\mathbf{a}, b)$ are either linear combinations or shifted linear combination of LWE samples, both of which are indistinguishable from the uniform distribution. This is true because shifting by any fixed length preserves the shape of a distribution. Therefore, an efficient algorithm that can correctly guess the encrypted bit would be able to distinguish LWE samples from uniformly distributed samples. This allows Regev *et al.* [70] to prove that:

Theorem 4. *Based on the hardness assumptions of several lattice problems, the LWE cryptosystem is secure against the chosen-plaintext attacks using both classical and quantum computers.*

When the error terms e_i's are introduced:

$$\begin{aligned} b - \langle \mathbf{a}, \mathbf{s} \rangle &= \sum_{i \in S} b_i + \left\lfloor \frac{q}{2} \right\rfloor r - \langle \sum_{i \in S} \mathbf{a}_i, \mathbf{s} \rangle \\ &= \sum_{i \in S} (\langle \mathbf{a}_i, \mathbf{s} \rangle + e_i) - \left\lfloor \frac{q}{2} \right\rfloor r - \langle \sum_{i \in S} \mathbf{a}_i, \mathbf{s} \rangle \\ &= \left\lfloor \frac{q}{2} \right\rfloor r - \sum_{i \in S} e_i, \end{aligned}$$

in which S is the set of non-zero coordinates in $\mathbf{x}$. For a decryption error to occur, the accumulated error $\sum_{i \in S} e_i$ must be greater than the decision threshold $\lfloor q/4 \rfloor$. The probability of the error is given by [63]:

$$\mathrm{Err}_{\mathrm{LWE}} \approx 2(1 - \Phi(\frac{q/4}{\alpha q \sqrt{m/2}/\sqrt{2\pi}})) = 2(1 - \Phi(\frac{\sqrt{\pi}}{2\alpha\sqrt{m}})),$$

in which $\Phi(\cdot)$ is the cumulative distribution function of the standard Gaussian variable. We later use this expression to find the practical parameters for the lattice PUF.

4.3 Design of Lattice PUF

A strong PUF is a function f that maps input challenges $\mathbf{c}$ to output responses r in such a way that its physical instantiations are unique, robust, and show good randomness. We now show how to realize a PUF whose challenge-response behavior is defined by the decryption function of the LWE cryptosystem so that its ML resistance is guaranteed. Such a PUF is achieved by implementing the LWE decryption function using a POK-derived

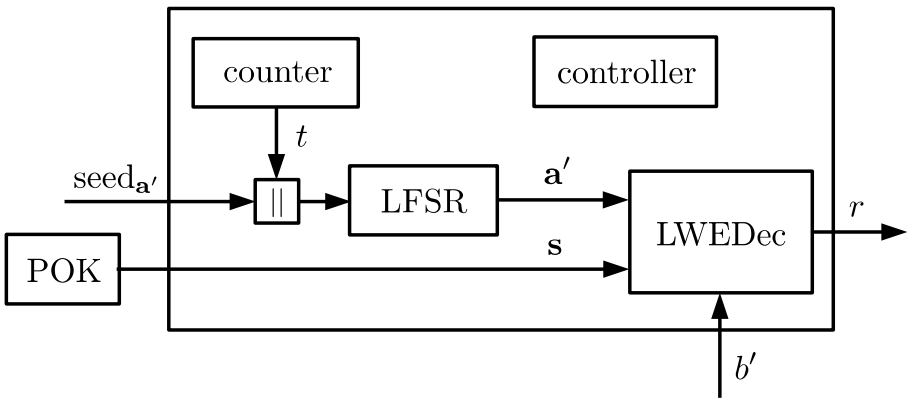


Figure 4.1: Top-level architecture and data flow of the lattice PUF.

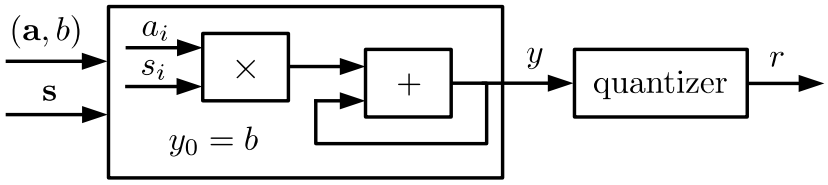


Figure 4.2: Architecture of LWE decryption function.

secret as the private key. In such a PUF, ciphertexts and decrypted plaintext bits are treated as PUF challenges and responses respectively. However, such a direct implementation results in a very large challenge word, making challenge-transfer costs prohibitive. We overcome this problem *by exploiting distributional relaxations allowed by recent work in space-efficient LWEs*, dramatically reducing the challenge transfer cost.

The top-level architecture of the proposed lattice PUF is shown in Figure 4.1.

4.3.1 LWE Decryption Function

Figure 4.2 shows the core component of the proposed lattice PUF: the LWE decryption function. It takes a binary challenge vector $\mathbf{c} = \{c_0, c_1, \ldots, c_{N-1}\}$ of size $N = (n+1)\log q$ which maps to a ciphertext $(\mathbf{a}, b)$ in the following way:

$$a_i = \sum_{j=0}^{\log q - 1} c_{(i-1)\log q + j} 2^j, \ \forall i \in \{1, 2, \ldots, n\},$$

$$b = \sum_{j=0}^{\log q - 1} c_{n \log q + j} 2^j.$$

Here a_i denotes the i-th element of the integer vector $\mathbf{a} \in \mathbb{Z}_q^n$. In this paper, without specification, $\log(x)$ refers to $\log_2(x)$. Similarly, the private key $\mathbf{s}$ for the corresponding LWE decryption function is realized by a binary secret key $\mathbf{W} = \{W_0, W_1, \ldots, W_{n \log q - 1}\}$ of size $n \log q$:

$$s_i = \sum_{j=0}^{\log q - 1} W_{(i-1)\log q + j} 2^j, \ \forall i \in \{1, 2, \ldots, n\}.$$

A modulo-dot-product $b - \langle \mathbf{a}, \mathbf{s} \rangle$ is computed using the modulo-multiply-accumulate unit. It can be implemented in a serial way using n stages. Recall that all additions and multiplications are performed in modulo q. Since q is a power of 2 in our construction, modulo addition and multiplication can be naturally implemented by integer addition and multiplication that keep only the last $\log q$-bit result. Finally the response r is produced by a quantization operation $r = Q(b - \langle \mathbf{a}, \mathbf{s} \rangle)$:

$$Q(x) = \begin{cases} 0 & x \in [0, \frac{q}{4}] \cup (\frac{3q}{4}, q-1], \\ 1 & x \in (\frac{q}{4}, \frac{3q}{4}]. \end{cases}$$

The computation above can be directly implemented as a strong PUF with 2^N CRPs since it maps a challenge vector $\mathbf{c} \in \{0,1\}^N$ into a binary response $r \in \{0, 1\}$. We now discuss parameter selection for the LWE decryption

function. In general, we seek to find design parameters such that (1) the resulting PUF has excellent statistical properties, such as uniformity, uniqueness, and reliability, (2) successful ML attacks against it require an un-affordably high time complexity in practice, and (3) its hardware implementation costs are minimized.

Prior theoretical arguments establish the impossibility of a polynomial-time attacker. To guarantee practical security, we need to estimate the number of samples and the actual running time (or a number of CPU operations) required for a successful ML attack. Regev *et al.* shows that a small number of samples are enough to solve an LWE problem, but in an exponential time [70]. Thus, we refer to runtime as concrete ML resistance (or ML hardness) and say that a PUF has k-bit ML resistance if any successful ML attack requires at least 2^k operations. We adopt the estimator developed by Albrecht *et al.* [3] to estimate concrete ML hardness. The concrete hardness of an LWE problem increases with the increase of LWE parameters n, q, and α for all types of attacks. Recall that n represents the lattice dimension, q represents the range of integer for each dimension, and α reflects the noise level in CRP (ciphertext) generation. For a given set of parameters, the estimator compares the complexity of several most effective attacks, including decoding, basis reduction, and meet-in-the-middle attacks [16, 36, 52]. We utilize the estimator in a black-box fashion to find the set of parameters with the target of 128-bit concrete ML resistance.

We consider two metrics of implementation cost, both of which scale with n: the number of challenge and secret bits needed ($n \log q$), and the number of multiply-accumulate (MAC) operations (n). This motivates the need to decrease n.

For conventional PUFs, such as arbiter PUF and SRAM PUF, an output error is due to environmental noise, e.g. delay changes in arbiter PUF and FET strength changes in SRAM PUF with both voltage and temperature. In contrast, output errors of the lattice PUF come from two sources: (1) environmental errors of secret bits, and (2) errors of decryption during response generation. The former can be thought as the failure of key reconstruction in POKs. Since a single bit-flip completely changes the challenge-response behavior of LWE decryption function, the failure rate of key reconstruction needs to be low, e.g. 10^{-6} (as widely adopted in other PUF applications [56]). Section 4.4 describes how the target failure rate can be achieved via a conventional FE based on the error-correcting codes. The latter corresponds to the decryption error and is orthogonal to errors in the secret key $\mathbf{s}$. Recall that in CRP generation of the lattice PUF, a bit of plaintext r is sampled and the ciphertext $\mathbf{c}$ is produced by a noisy encryption function $\mathbf{c} = \mathrm{Enc}(r)$. Given ciphertext $\mathbf{c}$ as input challenge, the decryption function can output a wrong response $r' \neq r$ when the accumulated error $\sum_{i \in S} e_i$ in the encryption function exceeds the decision boundary.

The model for evaluating the decryption error rate is shown in Section 4.2. In order for a strong PUF to be used in direct authentication, its decryption error rate should be small enough for reliable distinguishability of long strings. Following the work of Yu *et al.* [93], we set the target at 5%. Figure 4.3 explores the trade-off between the number of secret bits and the decryption error rate needed for 128-bit concrete ML hardness. It shows that, at fixed concrete ML hardness, the decryption error rate decreases super exponentially with the number of secret bits.

Considering the design metrics above, a feasible set of parameters is

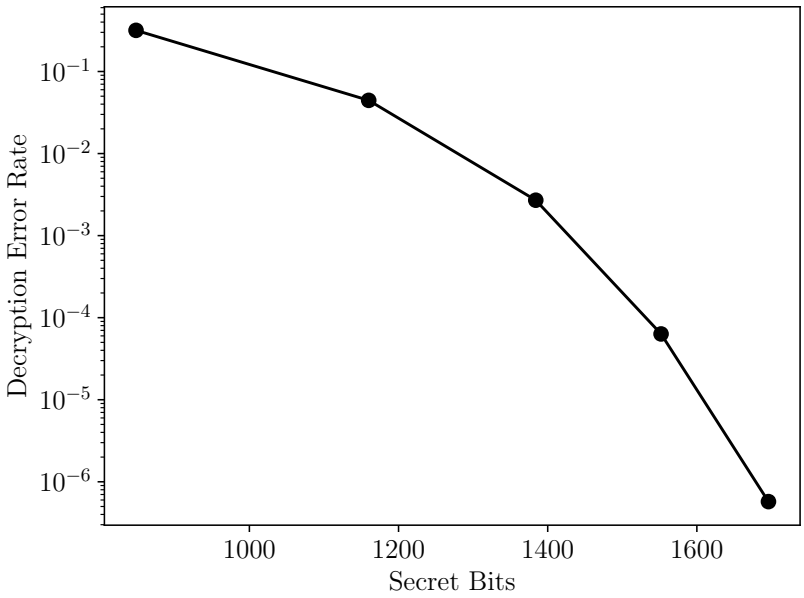


Figure 4.3: Super-exponential decrease of decryption error rate with the increase of secret bits. The analysis is done for 128-bit concrete hardness.

found using the estimator [3]. By setting $n = 145$, $q = 256$, $m = 256$ and $\alpha = 2.77\%$, we achieve a lattice PUF with 128-bit concrete hardness and a decryption error rate of 4.55%.

In order to get a 1-bit response, $(n + 1) \log q = 1168$ bits need to be sent to the lattice PUF as a challenge. For direct authentication applications, usually around 100 bits of responses are required. Therefore, the direct implementation described so far would require $C = 116.8K$ challenge bits. This high ratio of challenge length to response length limits its practical use in many scenarios when communication is expensive. We next describe an improved design that overcomes this limitation.

4.3.2 Challenge Compression through Distributional Relaxation

We introduce a challenge compression method that employs a suitable distributional relaxation to improve the efficiency of challenge transfer. The basic design described in the previous section requires a challenge $\mathbf{c}$ in the form $\mathbf{c} = (\mathbf{a}, b)$ to be sent to the PUF. To represent vector $\mathbf{a} \in \mathbb{Z}_q^n$ requires $n \log q$ bits while to represent scalar $b \in \mathbb{Z}_q$ requires only $\log q$ bits. Thus, the major cost of transmission is in sending $\mathbf{a}$. We wish to avoid sending $\mathbf{a}$ directly and, instead, to send a compressed (shorter) version of $\mathbf{a}$ and re-generate its full-size version on the PUF. Our approach is enabled by the recent results on the distributional behavior of $\mathbf{a} = \mathbf{A}^T\mathbf{x}$ [2] and the concept of space-efficient LWE [25]. We describe these two results below and show that the improved PUF design is still able to maintain concrete ML hardness against practical attacks.

Recall that b is given by:

$$b = \mathbf{b}^T\mathbf{x} + r\,\lfloor q/2 \rfloor = (\mathbf{A}\mathbf{s} + \mathbf{e})^T\mathbf{x} + r\,\lfloor q/2 \rfloor = (\mathbf{A}^T\mathbf{x})^T\mathbf{s} + \mathbf{e}^T\mathbf{x} + r\,\lfloor q/2 \rfloor .$$

First, we show that the term $\mathbf{a} = \mathbf{A}^T\mathbf{x}$, in the challenge, can be replaced by a uniformly random vector $\mathbf{a}^* \in \mathbb{Z}_q^n$. That allows us to represent challenge $\mathbf{c} = (\mathbf{a}, b)$ as $\mathbf{c}^* = (\mathbf{a}^*, b^*)$ in the following way:

$$\begin{cases} \mathbf{a} = \mathbf{A}^T\mathbf{x} \\ b = (\mathbf{A}^T\mathbf{x})^T\mathbf{s} + \mathbf{e}^T\mathbf{x} + r\,\lfloor q/2 \rfloor \end{cases} \rightarrow \begin{cases} \mathbf{a}^* \\ b^* = \mathbf{a}^{*T}\mathbf{s} + \mathbf{e}^T\mathbf{x} + r\,\lfloor q/2 \rfloor \end{cases} .$$

Akavia *et al.* [2] proved: (1) the true ciphertext distribution $\mathbf{c} = (\mathbf{a}, b)$ is *statistically* close to $(\mathbf{a}^*, b^*)$; (2) such statistical closeness ensures that the ML hardness of the lattice PUF is maintained.

Second, we show that the uniformly random $\mathbf{a}^*$ can be approximated by a PRNG-generated $\mathbf{a}'$, so that the PUF design allows for sending $(\text{seed}_{\mathbf{a}'}, b')$

as challenge. Here $\text{seed}_{\mathbf{a}'}$ is the seed of a lightweight PRNG, such as an LFSR, needed to re-generate the n-dimensional vector $\mathbf{a}'$. b' follows as

$$b' = (\mathbf{a}')^T\mathbf{s} + \mathbf{e}^T\mathbf{x} + r\,\lfloor q/2 \rfloor = \text{LFSR}(\text{seed}_{\mathbf{a}'})^T\mathbf{s} + \mathbf{e}^T\mathbf{x} + r\,\lfloor q/2 \rfloor\,.$$

Several state-of-the-art LWE cryptosystems adopt the theory by Galbraith *et al.* [25] and use efficient PRNGs to generate public values. For example, TESLA uses the ChaCha8 stream cipher [4], NewHope [5] and Frodo [12] propose to use AES-128 or SHAKE-128. We find that a more lightweight PRNG is needed for a resource-constrained device. Galbraith *et al.* showed that it is not necessary to use strong PRNGs based on block ciphers or hash functions [25]. The desired properties of a lightweight PRNG are: (1) it requires only a few clock cycles to generate each public value; (2) on-device storage requirements are low. We find that the LFSR is a suitable PRNG. Figure 4.4 shows the structure of a k-bit LFSR. A k-bit LFSR is composed of k serially connected registers and a small number of XOR gates. A k-bit seed is first fed into the registers as the initial state. On a clock edge, the register bits propagate one position to the left. Feedback computes the XORed value of some bits and writes them into the right most bit. Figure 4.5 shows a lightweight FPGA implementation of a 256-bit LFSR. The 256 register cells are realized using 16 on-chip registers and one shift register LUT (SRL) with 1-bit data width and 240-bit depth. We use the LFSR to generate multiple $\log q$-bit $\mathbf{a}'$.

The security of the new scheme is guaranteed because $\mathbf{a}'$ maintains needed properties to resist the standard attacks on LWE such as CVP reduction, decoding, and basis reduction, as proven by Galbraith *et al.* [25]. These properties include:

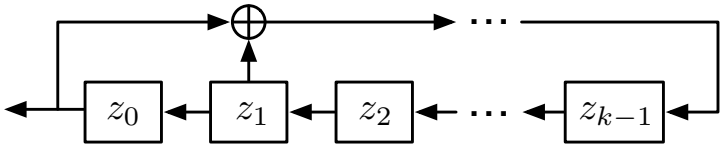


Figure 4.4: Construction of a k-bit LFSR.

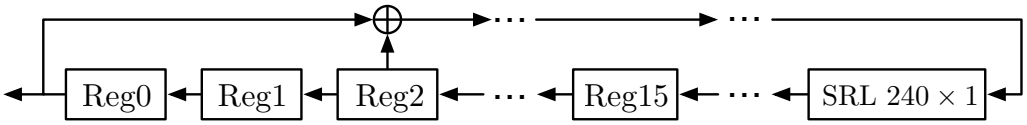


Figure 4.5: FPGA implementation of a 256-bit LFSR.

- it is hard to find "nice" bases for a lattice with basis from LFSR-generated $\mathbf{a}'$;
- given an arbitrary vector in $\mathbb{Z}_q^n$, it is hard to represent it as a binary linear combination of LFSR-generated $\mathbf{a}'$'s;
- it is hard to find a short vector $\mathbf{w}$ that is orthogonal to LFSR-generated $\mathbf{a}'$'s.

The ability to rely on an LFSR to produce $\mathbf{a}'$ allows a dramatic reduction in challenge transfer cost. We developed the lattice PUF scheme with LFSR, as shown in Figure 4.6a. With LWE parameters chosen as Section 4.3.1, using a seed of length 256 is able to reduce the challenge length from 1168 to $256 + 8 = 264$ per one bit of response. The improvement of efficiency becomes more pronounced for generating multiple responses: This is because $\mathbf{a}'_1 \dots \mathbf{a}'_t$ can be generated sequentially from the l-bit seed, so that only the seed and $b'_1, \dots, b'_t \in \mathbb{Z}_q$ are required to be sent to the PUF side. 100 bits of responses

now require only transmitting $128 + 100 \times \log 256 = 928$ bits for challenges, compared to $100 \times (145 + 1) \log 256 = 116.8$ Kbits without the LFSR. The challenge transfer cost is reduced by about 100X.

4.3.3 Countermeasure for Active Attack

In this section, we introduce a simple defense to protect our PUF against a standard attack on the LWE decryption function. The attack is premised on the ability to supply arbitrary challenges (ciphertexts) as inputs to the decryption function. The attack proceeds as follows. The attacker fixes $\mathbf{a}$ and enumerates all possible $b \in \mathbb{Z}_q$ for challenge $\mathbf{c} = (\mathbf{a}, b)$. As b increases from 0 to $q-1$, the response $r = Q(b - \langle \mathbf{a}, \mathbf{b} \rangle)$ changes from $Q(b - \langle \mathbf{a}, \mathbf{s} \rangle) = 0$ to $Q(b + 1 - \langle \mathbf{a}, \mathbf{s} \rangle) = 1$ exactly when b satisfies

$$b - \langle \mathbf{a}, \mathbf{s} \rangle = q/4.$$

We denote this specific value of b as $\hat{b}$. The exact value of $\langle \mathbf{a}, \mathbf{s} \rangle$ can then be extracted by $\langle \mathbf{a}, \mathbf{s} \rangle = \hat{b} - q/4$. By repeating this procedure n times, the attacker is able to set up n linear equations (without errors):

$$\begin{aligned} \langle \mathbf{a}_0, \mathbf{s} \rangle &= \hat{b}_0 - q/4, \\ \langle \mathbf{a}_1, \mathbf{s} \rangle &= \hat{b}_1 - q/4, \\ &\cdots \\ \langle \mathbf{a}_{n-1}, \mathbf{s} \rangle &= \hat{b}_{n-1} - q/4. \end{aligned}$$

Gaussian elimination can then be used to solve for $\mathbf{s}$. The reason the attack succeeds is that attackers are able to fix $\mathbf{a}$ and use it for multiple values of b. For the variant with the LFSR, shown in Figure 4.6a, this standard attack still applies: the attacker now fixes $\text{seed}_{\mathbf{a}'}$, calculates $\mathbf{a}'_i$ values from the LFSR, and uses the same procedure to extract $\mathbf{s}$.

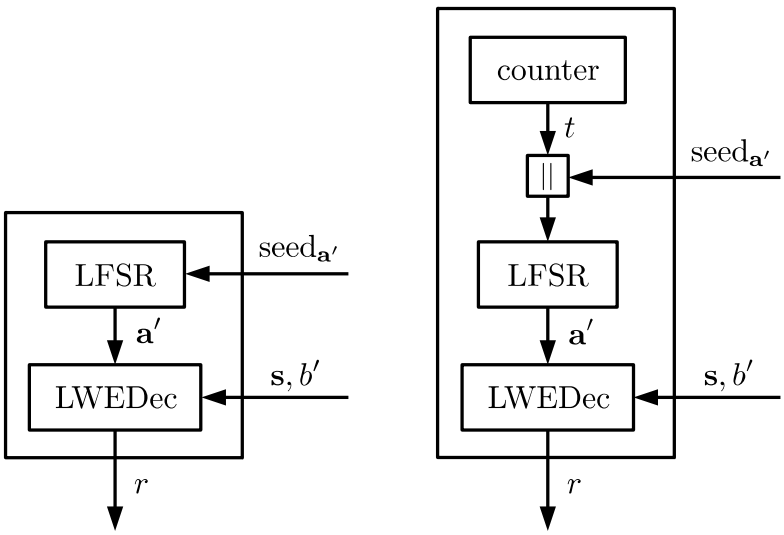


Figure 4.6: (a) Basic scheme and (b) counter-based scheme of challenge generation with LFSR.

Preventing this attack requires that the attacker is not able to arbitrarily choose the PUF challenge. This can be achieved by embedding a self-incrementing counter into the system design, in a way similar to the work of Yu *et al.* [93]. As shown in Figure 4.6b, the concatenation of the counter value t and the challenger-provided seed$_{\mathbf{a}'}$ becomes the seed of LFSR to generate $\mathbf{a}'$. The counter value is public and increments by 1 for each response generation. In this way, the attacker cannot enumerate all b' values while keeping $\mathbf{a}'$ unchanged. Therefore, it is infeasible to derive the linear equations and solve for $\mathbf{s}$. (Note that we cannot use the XORed value of t and seed$_{\mathbf{a}'}$, which is another possibility, as the seed of LFSR. The reason is that if the PUF uses $t \oplus$ seed$_{\mathbf{a}'}$ as the seed, the attacker can control the value of seed$_{\mathbf{a}'}$ to make the seed constant. For example, let us assume that the attacker sends seed$_{\mathbf{a}'}$, when the counter value is t_1. When the counter value becomes t_2, the attacker

can send $\text{seed}_{\mathbf{a}'} \oplus t_1 \oplus t_2$ to keep the seed unchanged.)

4.4 Experimental Results

In this section we evaluate statistical properties of the lattice PUF, including uniformity, uniqueness, and reliability with parameters chosen in Section 4.3. We also present the implementation cost on the FPGA platform and compare it with prior work.

4.4.1 Statistical Analysis

Uniformity of a PUF characterizes unbiasedness, namely, the proportion of '0's and '1's in the output responses. For an ideal PUF f, the proportion needs to be 50%. We adopt the definition of uniformity by Maiti *et al.* [58] based on the average Hamming weight $\text{HW}(f)$ of responses $\mathbf{r}$ to randomly sampled challenges $\mathbf{c}$'s:

$$\text{HW}(f) = \mathbf{E}_{\mathbf{c}}[\text{HW}(\mathbf{r})] = \mathbf{E}_{\mathbf{c}}[\text{HW}(f(\mathbf{c}))].$$

Here $\mathbf{E}_X$ represents expectation over random variable X. Note that $\mathbf{c}$ follows the ciphertext distribution rather than the usual uniform distribution [58]. Figure 4.7 shows uniformity obtained using 1000 randomly selected challenges. The distribution is centered at 49.92%, the standard deviation is 1.58%.

Uniqueness measures the ability of a PUF to be uniquely distinguished among a set of PUFs. We define this metric to be the average inter-class HD of responses $(\mathbf{r}_i, \mathbf{r}_j)$ under the same challenges $\mathbf{c}$ for a randomly picked PUF pair (f_i, f_j) [58]:

$$\text{HD}(f_i, f_j) = \mathbf{E}_{\mathbf{c}}[\text{HD}(\mathbf{r}_i, \mathbf{r}_j)] = \mathbf{E}_{\mathbf{c}}[\text{HD}(f_i(\mathbf{c}), f_j(\mathbf{c}))].$$

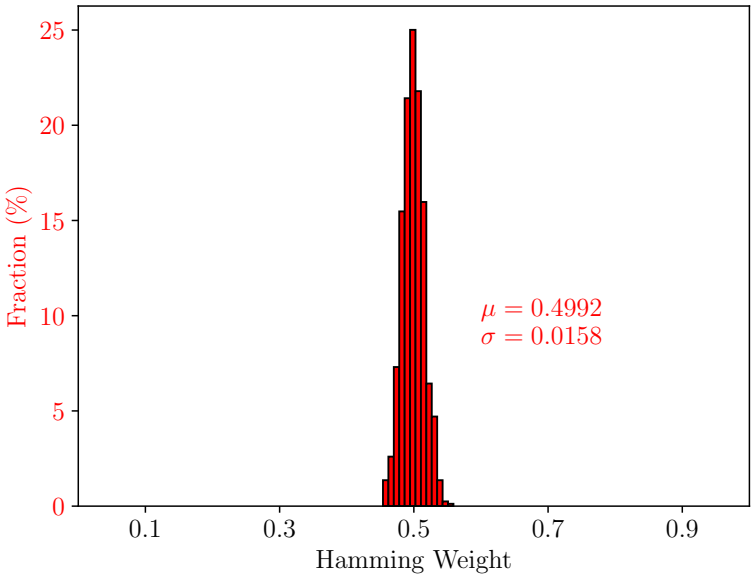


Figure 4.7: Uniformity of lattice PUF output.

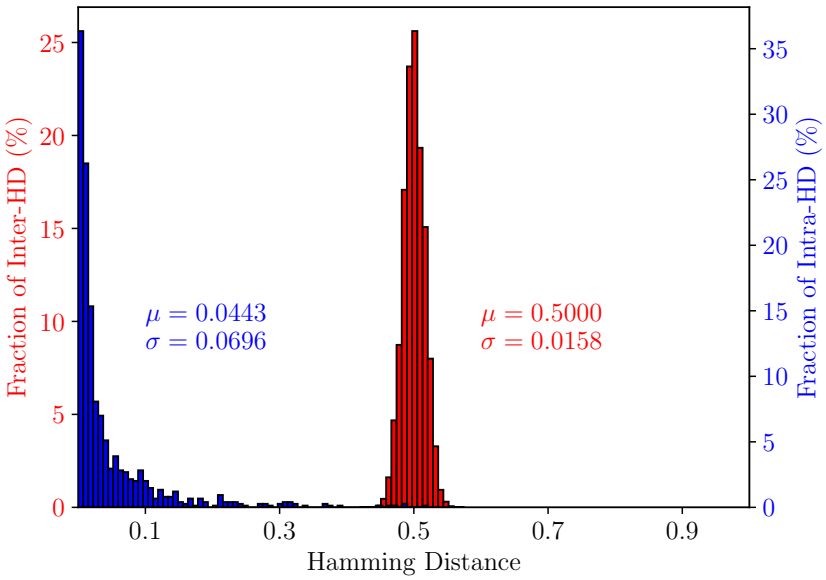


Figure 4.8: Uniqueness and reliability of lattice PUF.

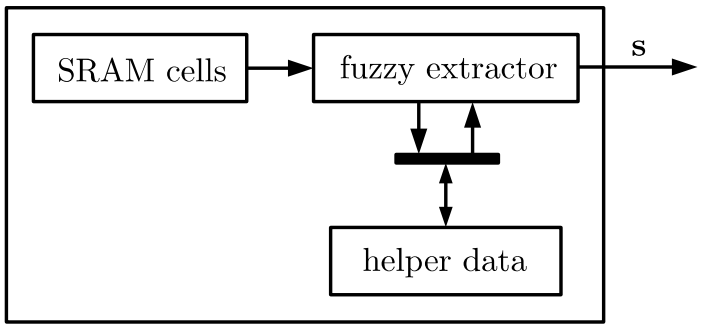


Figure 4.9: POK uses an FE to ensure stability of the secret key.

For ideal PUFs, responses under the same challenges are orthogonal, namely, $\mathrm{HD}(f_i, f_j)$'s are close to 50%. Uniqueness is also evaluated under the ciphertext distribution.

Uniqueness is shown in Figure 4.8, evaluated for 1000 PUF instances. The lattice PUF achieves near-optimal uniqueness: inter-class HD is centered at 50.00%, its standard deviation is 1.58%.

Reliability of a PUF f is characterized by the average BER of outputs with respect to their enrollment values [58]:

$$\mathrm{BER} = \mathbf{E}_{f'}[\mathrm{HD}(f, f')] = \mathbf{E}_{f',\mathbf{c}}[\mathrm{HD}(f(\mathbf{c}), f'(\mathbf{c}))].$$

As discussed in Section 4.3, the overall BER of the lattice PUF is due to two components: the failure rate of key reconstruction and LWE decryption error rate. Intra-class HD in Figure 4.8 reflects the result of decryption errors by assuming a perfect key reconstruction.

4.4.2 Hardware Implementation Results

We present the details of implementing the lattice PUF, shown in Figure 4.1:

Table 4.1: Configuration of error-correcting codes.

Raw BER (%)	**Error-Correcting Code**		**Raw POKs**
	Outer code	Inner code	
1	[236, 128, 14]	N/A	2,360
5	[218, 128, 11]	[3, 1, 1]	6,540
10	[220, 128, 12]	[5, 1, 2]	11,000
15	[244, 128, 15]	[7, 1, 3]	17,080

- The design was synthesized, configured, and tested on a Xilinx Spartan-6 FPGA (XC6SLX45), a low-end FPGA in 45nm technology.
- The core block implements the LWE decryption function (LWEDec includes an 8-bit MAC and a quantization block, as shown in Figure 4.2).
- A 256-bit LFSR is implemented using 1 SRL240 $\times$ 1, 16 registers, and 3 XOR gates, and is used to generate public $\mathbf{a}'$. The seed needs to be loaded only once to generate multiple response bits.
- A 128-bit self-incrementing counter is implemented. The 128-bit counter value t is public and stored in nonvolatile memory (NVM). The concatenation of t and the challenger-provided 128-bit $\text{seed}_{\mathbf{a}'}$ is used as the seed of the LFSR.
- A controller block controlls the operation flow of the lattice PUF.

We use SRAM cell power-up states as the raw POK bits, and use an FE to generate a secret key of 1160 bits with the failure rate of reconstruction targeted at 10^{-6}. As shown in Figure 4.9, the FE uses helper data to produce a reliable secret key $\mathbf{s}$. We adopt the homogeneous error assumption, i.e., all cells have the same BER [13]. Prior work shows that intrinsic BERs of the various

Table 4.2: (a) Area consumption and (b) runtime of our reference lattice PUF implementation on Spartan-6 FPGA.

(a)

Module	**Size [slices]**
LFSR	27
LWEDec	2
Controller	16
Total	45

(b)

Step	**Time [μs]**
Seed $\text{seed}_{\mathbf{a}'}||t$ load for LFSR	8
1-bit decryption from LWEDec	39
Total @ 33 *MHz*	47

POKs range from 0.1% [42] to 15% [55]. We study the POK designs under four levels of raw BER: 1%, 5%, 10%, and 15% to explore design costs. We use concatenated error-correcting codes, with a repetition code as the inner code, and a shortened BCH code as the outer code. Concatenated codes are typically more efficient than single codes in terms of code length and hardware cost [13]. Table 4.1 lists the configuration of error-correcting codes used at different BER levels. At the raw BER of 5%, $6.5K$ cells are needed to construct the secret $\mathbf{s}$ of length 1160 bits at the target failure rate 10^{-6}.

The total size of the lattice PUF (without FE) for the Spartan-6 platform is 45 slices, most of which is taken up by the LFSR and the controller. Table 4.2a shows the breakdown of resources needed to realize the various modules. The total latency (at 33.3MHz clock) to generate a 1-bit PUF response is 47μs, and the total time to generate a 100-bit PUF response is, approximately, $8\mu s + 100 \times 39\mu s \approx 3.9ms$ since seed loading is only executed once. Table 4.2b lists the latency of each step of response generation.

Table 4.3: Hardware implementation costs of strong PUFs.

Design	**Platform**	**PUF Logic [Slices]**
POK+AES [11]	Spartan 6	80
Controlled PUF [29]	Spartan 6	127
CFE-based PUF [32, 38]	Zynq-7000	9,825
Lattice PUF	Spartan 6	45

Table 4.4: Hardware utilization in FE design on Spartan-6 FPGA.

Raw BER	**Outer Code**			**Inner Code**			**Total**		
(%)	Reg	LUT	Slice	Reg	LUT	Slice	Reg	LUT	Slice
1	905	893	276	0	0	0	905	893	276
5	730	688	232	0	1	1	730	689	233
10	785	740	243	0	3	2	785	743	245
15	973	913	326	0	7	3	973	920	329

We compare the implementation cost of the lattice PUF against established strong PUF designs [11, 29, 38] in Table 4.3. The original strong PUF based on AES [11] is implemented as an ASIC. Here, we adopt the design by Chu *et al.* [17] as an FPGA alternative to estimate the implementation cost of AES. Notice that the design by Bhargava *et al.* [11] uses no error correction since it guarantees reliability via dark bit masking. Similarly, the FPGA implementation of SHA-3 [40] is adopted to estimate the cost of a hash function for the controlled PUF [29]. The FPGA utilization result of the strong PUF based on the computational FE (CFE) is shown via the number of LUTs presented by Jin *et al.* [38]. We estimate the corresponding slice count according to the Zynq-7000 SoC Data Sheet [90]. We find that the implementation cost of the lattice PUF (without FE) is cheaper than that of AES on POK, controlled PUF, and CFE-based PUF.

Detailed costs of error correction in POKs for the lattice PUF with

different raw BERs are presented in Table 4.4. Assuming raw POK BER of 5%, the FE design of the lattice PUF requires 233 slices. This is cheaper than the linear solver block used in the CFE-based strong PUF [32, 38] which requires $65,700$ LUTs and $16,425$ slices.

Bibliography

[1] Michel Abdalla, Sonia Belaïd, and Pierre-Alain Fouque. Leakage-resilient symmetric encryption via re-keying. In *CHES*, pages 471–488. Springer, 2013.

[2] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *Theory of Cryptography Conference*, pages 474–495. Springer, 2009.

[3] Martin R Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.

[4] Erdem Alkim, Nina Bindel, Johannes A Buchmann, Özgür Dagdelen, and Peter Schwabe. Tesla: Tightly-secure efficient signatures from standard lattices. *IACR Cryptology ePrint Archive*, 2015:755, 2015.

[5] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange-a new hope. In *USENIX Security Symposium*, 2016.

[6] Anastacia B Alvarez, Wenfeng Zhao, and Massimo Alioto. Static physically unclonable functions for secure chip identification with 1.9–5.8% native bit instability at 0.6–1 V and 15 fJ/bit in 65 nm. *IEEE Journal of Solid-State Circuits*, 51(3):763–775, 2016.

[7] Georg T Becker. The gap between promise and reality: On the insecurity of xor arbiter pufs. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 535–555. Springer, 2015.

[8] Georg T Becker, Raghavan Kumar, et al. Active and passive side-channel attacks on delay based puf designs. *IACR Cryptology ePrint Archive*, 2014:287, 2014.

[9] Daniel J Bernstein. Introduction to post-quantum cryptography. In *Post-quantum cryptography*, pages 1–14. Springer, 2009.

[10] Mudit Bhargava and Ken Mai. An efficient reliable PUF-based cryptographic key generator in 65nm CMOS. In *Proceedings of the conference on Design, Automation & Test in Europe*, page 70. European Design and Automation Association, 2014.

[11] Mudit Bhargava and Ken Mai. An efficient reliable puf-based cryptographic key generator in 65nm cmos. In *Proceedings of the conference on Design, Automation & Test in Europe*, page 70. European Design and Automation Association, 2014.

[12] Joppe Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the ring! practical, quantum-secure key exchange from lwe. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1006–1018. ACM, 2016.

[13] Christoph Bösch, Jorge Guajardo, Ahmad-Reza Sadeghi, Jamshid Shokrollahi, and Pim Tuyls. Efficient helper data key extractor on fpgas. In

International Workshop on Cryptographic Hardware and Embedded Systems, pages 181–197. Springer, 2008.

[14] Wenjie Che, Mitchell Martin, Goutham Pocklassery, Venkata K Kajuluri, Fareena Saqib, and Jim Plusquellic. A privacy-preserving, mutual puf-based authentication protocol. *Cryptography*, 1(1):3, 2016.

[15] Qingqing Chen, György Csaba, Paolo Lugli, Ulf Schlichtmann, and Ulrich Rührmair. The bistable ring puf: A new architecture for strong physical unclonable functions. In *2011 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 134–141. IEEE, 2011.

[16] Yuanmi Chen and Phong Q Nguyen. Bkz 2.0: Better lattice security estimates. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 1–20. Springer, 2011.

[17] Junfeng Chu and Mohammed Benaissa. Low area memory-free fpga implementation of the aes algorithm. In *Field Programmable Logic and Applications (FPL), 2012 22nd International Conference on*, pages 623–626. IEEE, 2012.

[18] Jeroen Delvaux. *Security Analysis of PUF-Based Key Generation and Entity Authentication*. PhD thesis, Shanghai Jiao Tong University, China, 2017.

[19] Jeroen Delvaux, Dawu Gu, Ingrid Verbauwhede, Matthias Hiller, and Meng-Day Mandel Yu. Efficient fuzzy extraction of puf-induced secrets: Theory and applications. In *CHES*, pages 412–431. Springer, 2016.

[20] Christoph Dobraunig, Maria Eichlseder, Thomas Korak, and Florian Mendel. Side-channel analysis of keymill. In *COSADE*, pages 138–152. Springer, 2017.

[21] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *ICTACIS*, pages 523–540. Springer, 2004.

[22] Pedro Domingos and Michael Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine learning*, 29(2-3):103–130, 1997.

[23] Stefan Dziembowski, Sebastian Faust, Gottfried Herold, Anthony Journault, Daniel Masny, and François-Xavier Standaert. Towards sound fresh re-keying with hard (physical) learning problems. In *CRYPTO*, pages 272–301. Springer, 2016.

[24] Pierre-Alain Fouque, Gaëtan Leurent, Denis Réal, and Frédéric Valette. Practical electromagnetic template attack on hmac. In *CHES*, volume 5747, pages 66–80. Springer, 2009.

[25] Steven D Galbraith. Space-efficient variants of cryptosystems based on learning with errors. *url: https://www. math. auckland. ac. nz/˜ sgal018/compact-LWE. pdf*, 2013.

[26] Fatemeh Ganji, Shahin Tajik, Fabian Fäßler, and Jean-Pierre Seifert. Strong machine learning attack against pufs with no mathematical model. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 391–411. Springer, 2016.

[27] Fatemeh Ganji, Shahin Tajik, and Jean-Pierre Seifert. Pac learning of arbiter pufs. *Journal of Cryptographic Engineering*, 6(3):249–258, 2016.

[28] Blaise Gassend, Dwaine Clarke, Marten Van Dijk, and Srinivas Devadas. Silicon physical random functions. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 148–160. ACM, 2002.

[29] Blaise Gassend, Marten Van Dijk, Dwaine Clarke, Emina Torlak, Srinivas Devadas, and Pim Tuyls. Controlled physical random functions and applications. *ACM Transactions on Information and System Security (TISSEC)*, 10(4):3, 2008.

[30] Louis Goubin and Jacques Patarin. Des and differential power analysis the duplication method. In *CHES*, pages 728–728. Springer, 1999.

[31] Nikolaus Hansen. CMA-ES, covariance matrix adaptation evolution strategy for non-linear numerical optimization in python. `https://pypi.python.org/pypi/cma`, 2017.

[32] Charles Herder, Ling Ren, Marten van Dijk, Meng-Day Yu, and Srinivas Devadas. Trapdoor computational fuzzy extractors and stateless cryptographically-secure physical unclonable functions. *IEEE Transactions on Dependable and Secure Computing*, 14(1):65–82, 2017.

[33] Daniel E Holcomb, Wayne P Burleson, and Kevin Fu. Power-up SRAM state as an identifying fingerprint and source of true random numbers. *IEEE Transactions on Computers*, 58(9):1198–1210, 2009.

[34] Gabriel Hospodar, Roel Maes, and Ingrid Verbauwhede. Machine learning attacks on 65nm Arbiter PUFs: Accurate modeling poses strict bounds

on usability. In *2012 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 37–42. IEEE, 2012.

[35] Gabriel Hospodar, Roel Maes, and Ingrid Verbauwhede. Machine learning attacks on 65nm arbiter pufs: Accurate modeling poses strict bounds on usability. In *Information Forensics and Security (WIFS), 2012 IEEE International Workshop on*, pages 37–42. IEEE, 2012.

[36] Nick Howgrave-Graham. A hybrid lattice-reduction and meet-in-the-middle attack against ntru. In *Annual International Cryptology Conference*, pages 150–169. Springer, 2007.

[37] Supreet Jeloka, Kaiyuan Yang, Michael Orshansky, Dennis Sylvester, and David Blaauw. A sequence dependent challenge-response puf using 28nm sram 6t bit cell. In *2017 Symposium on VLSI Circuits*, pages C270–C271. IEEE, 2017.

[38] Chenglu Jin, Charles Herder, Ling Ren, Phuong Ha Nguyen, Benjamin Fuller, Srinivas Devadas, and Marten van Dijk. Fpga implementation of a cryptographically-secure puf based on learning parity with noise. *Cryptography*, 1(3):23, 2017.

[39] Mukund Kalyanaraman and Michael Orshansky. Novel strong PUF based on nonlinearity of MOSFET subthreshold operation. In *2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 13–18. IEEE, 2013.

[40] Jens-Peter Kaps, Panasayya Yalla, Kishore Kumar Surapathi, Bilal Habib, Susheel Vadlamudi, Smriti Gurung, and John Pham. Lightweight imple-

mentations of sha-3 candidates on fpgas. In *International Conference on Cryptology in India*, pages 270–289. Springer, 2011.

[41] Deniz Karakoyunlu and Berk Sunar. Differential template attacks on puf enabled cryptographic devices. In *WIFS*, pages 1–6. IEEE, 2010.

[42] Bohdan Karpinskyy, Yongki Lee, Yunhyeok Choi, Yongsoo Kim, Mijung Noh, and Sanghyun Lee. 8.7 physically unclonable function for secure key generation with a key error rate of 2e-38 in 45nm smart-card chips. In *Solid-State Circuits Conference (ISSCC), 2016 IEEE International*, pages 158–160. IEEE, 2016.

[43] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. CRC press, 2014.

[44] Michael Kearns and Leslie Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM (JACM)*, 41(1):67–95, 1994.

[45] Michael Kharitonov. Cryptographic hardness of distribution-specific learning. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 372–381. ACM, 1993.

[46] Adam R Klivans and Alexander A Sherstov. Cryptographic hardness for learning intersections of halfspaces. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 553–562. IEEE, 2006.

[47] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *CRYPTO*, pages 789–789. Springer, 1999.

[48] Joonho Kong, Farinaz Koushanfar, Praveen K Pendyala, Ahmad-Reza Sadeghi, and Christian Wachsmann. PUFatt: Embedded platform attestation based on novel processor-based PUFs. In *Proceedings of the 51st Annual Design Automation Conference*, pages 1–6. ACM, 2014.

[49] Raghavan Kumar and Wayne Burleson. On design of a highly secure puf based on non-linear current mirrors. In *Hardware-Oriented Security and Trust (HOST), 2014 IEEE International Symposium on*, pages 38–43. IEEE, 2014.

[50] Jae W Lee, Daihyun Lim, Blaise Gassend, G Edward Suh, Marten Van Dijk, and Srinivas Devadas. A technique to build a secret key in integrated circuits for identification and authentication applications. In *2004 Symposium on VLSI Circuits, Digest of Technical Papers*, pages 176–179. IEEE, 2004.

[51] Jae W Lee, Daihyun Lim, Blaise Gassend, G Edward Suh, Marten Van Dijk, and Srinivas Devadas. A technique to build a secret key in integrated circuits for identification and authentication applications. In *VLSI Circuits, 2004. Digest of Technical Papers. 2004 Symposium on*, pages 176–179. IEEE, 2004.

[52] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for lwe-based encryption. In *Cryptographers Track at the RSA Conference*, pages 319–339. Springer, 2011.

[53] Roel Maes. Physically unclonable functions: Constructions, properties and applications. 2012.

[54] Roel Maes, Vladimir Rozic, Ingrid Verbauwhede, Patrick Koeberl, Erik Van der Sluis, and Vincent van der Leest. Experimental evaluation of Physically Unclonable Functions in 65 nm CMOS. In *2012 Proceedings of the ESSCIRC (ESSCIRC)*, pages 486–489. IEEE, 2012.

[55] Roel Maes, Pim Tuyls, and Ingrid Verbauwhede. A soft decision helper data algorithm for sram pufs. In *Information Theory, 2009. ISIT 2009. IEEE International Symposium on*, pages 2101–2105. IEEE, 2009.

[56] Roel Maes, Anthony Van Herrewege, and Ingrid Verbauwhede. Pufky: A fully functional puf-based cryptographic key generator. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 302–319. Springer, 2012.

[57] Abhranil Maiti, Vikash Gunreddy, and Patrick Schaumont. A systematic method to evaluate and compare the performance of physical unclonable functions. In *Embedded systems design with FPGAs*, pages 245–267. Springer, 2013.

[58] Abhranil Maiti, Vikash Gunreddy, and Patrick Schaumont. A systematic method to evaluate and compare the performance of physical unclonable functions. In *Embedded systems design with FPGAs*, pages 245–267. Springer, 2013.

[59] Mehrdad Majzoobi, Farinaz Koushanfar, and Miodrag Potkonjak. Lightweight secure pufs. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD 2008)*, pages 670–673. IEEE, 2008.

[60] Mehrdad Majzoobi, Farinaz Koushanfar, and Miodrag Potkonjak. Lightweight secure pufs. In *Computer-Aided Design, 2008. ICCAD 2008. IEEE/ACM*

International Conference on, pages 670–673. IEEE, 2008.

[61] Sanu K Mathew, Sudhir K Satpathy, Mark A Anders, Himanshu Kaul, Steven K Hsu, Amit Agarwal, Gregory K Chen, Rachael J Parker, Ram K Krishnamurthy, and Vivek De. A 0.19 pJ/b PVT-variation-tolerant hybrid physically unclonable function circuit for 100% stable secure key generation in 22nm CMOS. In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 278–279. IEEE, 2014.

[62] Marcel Medwed, François-Xavier Standaert, Johann Großschädl, and Francesco Regazzoni. Fresh re-keying: Security against side-channel and fault attacks for low-cost devices. *AFRICACRYPT*, 6055:279–296, 2010.

[63] Daniele Micciancio and Oded Regev. Lattice-based cryptography. In *Post-quantum cryptography*, pages 147–191. Springer, 2009.

[64] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2012.

[65] Michael Orshansky, Sani Nassif, and Duane Boning. *Design for manufacturability and statistical design: a constructive approach*. Springer Science & Business Media, 2007.

[66] Olivier Pereira, François-Xavier Standaert, and Srinivas Vivek. Leakage-resilient authentication and encryption from symmetric cryptographic primitives. In *CCS*, pages 96–108. ACM, 2015.

[67] Peter Pessl and Stefan Mangard. Enhancing side-channel analysis of binary-field multiplication with bit reliability. In *CT-RSA*, pages 255–270. Springer, 2016.

[68] Behzad Razavi. *Design of Analog CMOS Integrated Circuits*, pages 316–317. McGraw-Hill, 2001.

[69] Behzad Razavi. The StrongARM latch [a circuit for all seasons]. *IEEE Solid-State Circuits Magazine*, 7(2):12–17, 2015.

[70] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):34, 2009.

[71] Masoud Rostami, Mehrdad Majzoobi, Farinaz Koushanfar, Dan S Wallach, and Srinivas Devadas. Robust and reverse-engineering resilient puf authentication and key-exchange by substring matching. *TETC*, 2(1):37–49, 2014.

[72] Ulrich Rührmair, Frank Sehnke, Jan Sölter, Gideon Dror, Srinivas Devadas, and Jürgen Schmidhuber. Modeling attacks on physical unclonable functions. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 237–249. ACM, 2010.

[73] Ulrich Rührmair, Jan Sölter, Frank Sehnke, Xiaolin Xu, Ahmed Mahmoud, Vera Stoyanova, Gideon Dror, Jürgen Schmidhuber, Wayne Burleson, and Srinivas Devadas. PUF modeling attacks on simulated and silicon data. *IEEE Transactions on Information Forensics and Security*, 8(11):1876–1891, 2013.

[74] Sudhir Satpathy, Sanu Mathew, Jiangtao Li, Patrick Koeberl, Mark Anders, Himanshu Kaul, Gregory Chen, Amit Agarwal, Steven Hsu, and Ram Krishnamurthy. 13fj/bit probing-resilient 250k PUF array with soft darkbit masking for 1.94% bit-error in 22nm tri-gate CMOS. In

2014-40th European Solid State Circuits Conference (ESSCIRC), pages 239–242. IEEE, 2014.

[75] Ying Su, Jeremy Holleman, and Brian P Otis. A digital 1.6 pJ/bit chip identification circuit using process variations. *IEEE Journal of Solid-State Circuits*, 43(1):69–77, 2008.

[76] G Edward Suh and Srinivas Devadas. Physical unclonable functions for device authentication and secret key generation. In *Proceedings of the 44th annual design automation conference*, pages 9–14. ACM, 2007.

[77] G Edward Suh and Srinivas Devadas. Physical unclonable functions for device authentication and secret key generation. In *Proceedings of the 44th annual Design Automation Conference*, pages 9–14. ACM, 2007.

[78] Mostafa Taha, Arash Reyhani-Masoleh, and Patrick Schaumont. Stateless leakage resiliency from nlfsrs. In *HOST*, pages 56–61. IEEE, 2017.

[79] Mostafa Taha and Patrick Schaumont. Side-channel analysis of mac-keccak. In *HOST*, pages 125–130. IEEE, 2013.

[80] Mostafa MI Taha, Arash Reyhani-Masoleh, and Patrick Schaumont. Keymill: Side-channel resilient key generator. *IACR Cryptology ePrint Archive*, 2016:710, 2016.

[81] Yuan Taur, Douglas A Buchanan, Wei Chen, David J Frank, Khalid E Ismail, Shih-Hsien Lo, George A Sai-Halasz, Raman G Viswanathan, H-JC Wann, Shalom J Wind, et al. CMOS scaling into the nanometer regime. *Proceedings of the IEEE*, 85(4):486–504, 1997.

[82] Stefan Tillich, Christoph Herbst, and Stefan Mangard. Protecting aes software implementations on 32-bit processors against power analysis. In *ACNS*, pages 141–157. Springer, 2007.

[83] Kris Tiri, Moonmoon Akmal, and Ingrid Verbauwhede. A dynamic and differential cmos logic with signal independent power consumption to withstand differential power analysis on smart cards. In *ESSCIRC 2002*, pages 403–406. IEEE, 2002.

[84] Thomas Unterluggauer, Mario Werner, and Stefan Mangard. Side-channel plaintext-recovery attacks on leakage-resilient encryption. In *DATE*, pages 1318–1323. IEEE, 2017.

[85] Anthony Van Herrewege, Stefan Katzenbeisser, Roel Maes, Roel Peeters, Ahmad-Reza Sadeghi, Ingrid Verbauwhede, and Christian Wachsmann. Reverse fuzzy extractors: Enabling lightweight mutual authentication for puf-enabled rfids. In *Financial Cryptography*, volume 7397, pages 374–389. Springer, 2012.

[86] Arunkumar Vijayakumar, Vinay C Patil, Charles B Prado, and Sandip Kundu. Machine learning resistant strong puf: Possible or a pipe dream? In *Hardware Oriented Security and Trust (HOST), 2016 IEEE International Symposium on*, pages 19–24. IEEE, 2016.

[87] Alexander Wild, Amir Moradi, and Tim Güneysu. Evaluating the duplication of dual-rail precharge logics on fpgas. In *COSADE*, pages 81–94. Springer, 2015.

[88] Xiaodan Xi, Aydin Aysu, and Michael Orshansky. Fresh re-keying with strong pufs: A new approach to side-channel security. In *2018*

IEEE International Symposium on Hardware Oriented Security and Trust (HOST), pages 118–125. IEEE, 2018.

[89] Xiaodan Xi, Haoyu Zhuang, Nan Sun, and Michael Orshansky. Strong subthreshold current array puf with 2 65 challenge-response pairs resilient to machine learning attacks in 130nm cmos. In *VLSI Circuits, 2017 Symposium on*, pages C268–C269. IEEE, 2017.

[90] Xilinx. *Zynq-7000 SoC Data Sheet: Overview*, 7 2018. v1.11.1.

[91] Xiaolin Xu, Ulrich Rührmair, Daniel E Holcomb, and Wayne Burleson. Security evaluation and enhancement of bistable ring pufs. In *International Workshop on Radio Frequency Identification: Security and Privacy Issues*, pages 3–16. Springer, 2015.

[92] Kaiyuan Yang, Qing Dong, David Blaauw, and Dennis Sylvester. A physically unclonable function with BER $< 10^{-8}$ for robust chip authentication using oscillator collapse in 40nm CMOS. In *2015 IEEE International Solid-State Circuits Conference-(ISSCC)*, pages 1–3. IEEE, 2015.

[93] Meng-Day Yu, Matthias Hiller, Jeroen Delvaux, Richard Sowell, Srinivas Devadas, and Ingrid Verbauwhede. A lockdown technique to prevent machine learning on pufs for lightweight authentication. *IEEE Transactions on Multi-Scale Computing Systems*, 2(3):146–159, 2016.

[94] Liwei Zhang, A Adam Ding, Yunsi Fei, and Pei Luo. Efficient 2nd-order power analysis on masked devices utilizing multiple leakage. In *HOST*, pages 118–123. IEEE, 2015.